(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
24 April 2025 (24.04.2025)

WIPO | PCT

(10) International Publication Number
WO 2025/085455 A1

(51) International Patent Classification:
G06F 21/55 (2013.01) *H04L 9/40* (2022.01)
G06N 3/08 (2023.01) *G06F 16/2458* (2019.01)

(21) International Application Number:
PCT/US2024/051449

(22) International Filing Date:
15 October 2024 (15.10.2024)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
63/590,760 16 October 2023 (16.10.2023) US
63/561,509 05 March 2024 (05.03.2024) US

(71) Applicant: **SENTINELONE, INC.** [US/US]; 444 Castro Street, Suite 400, Mountain View, CA 94041 (US).

(72) Inventors: **STEWART, Gregor**; 444 Castro Street, Suite 400, Mountain View, CA 94041 (US). **KAHN, Ely**; 444 Castro Street, Suite 400, Mountain View, CA 94041 (US). **BINENFELD, Amir**; 444 Castro Street, Suite 400, Mountain View, CA 94041 (US).

(74) Agent: **FRANCIS, Jason** et al.; P.O. Box 1247, Seattle, WA 98111-1247 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST,

(54) Title: DELEGABLE AUTOMATED SYSTEM FOR REVIEW OF NETWORK THREATS

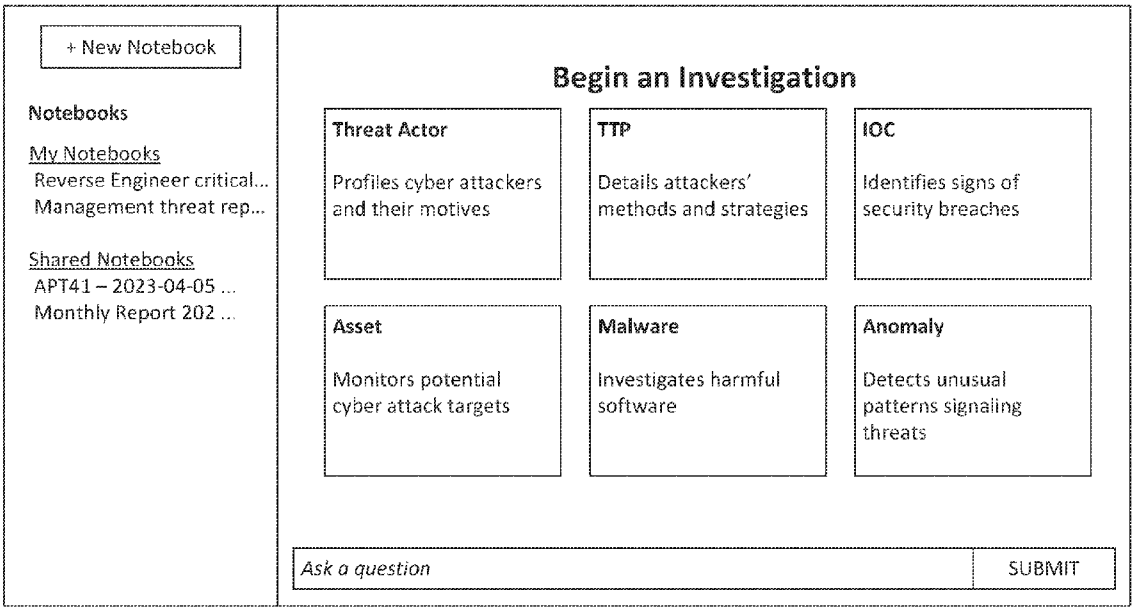

FIG. 3

(57) Abstract: This disclosure describes approaches for investigating events, such as a security alerts, which can be associated with malware, network threats, and so forth. Some embodiments relate to guided investigation, in which artificial intelligence/machine learning models are used to guide an analyst through an investigation. In some embodiments, a machine learning model utilizes a knowledge base to enable users to perform analysis without specific knowledge and/or with coding. In some embodiments, the approaches herein include determining actions to take in response to an analysis. In some embodiments, an investigation can be partially or fully automated.

WO 2025/085455 A1

[Continued on next page]

SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report (Art. 21(3))*

DELEGABLE AUTOMATED SYSTEM FOR REVIEW OF NETWORK THREATS

TECHNICAL FIELD

[0001] This disclosure generally relates to approaches for investigating events, such as a security alerts, which can be associated with malware, network threats, and so forth. Some embodiments relate to automated investigation. Some embodiments relate to guided investigation, in which machine learning is used to guide an analyst through an investigatory process.

BACKGROUND

[0002] The approaches described in this section are approaches that could be pursued, but not necessarily approaches that have been previously conceived or pursued. Thus, unless otherwise indicated, it should not be assumed that any of the material described in this section qualifies as prior art merely by virtue of its inclusion in this section.

[0003] There are many cyber security tools (e.g., software, services, and platforms) that operate to help protect computer systems and networks from potential attacks and network threats. These tools often generate security alerts associated with potential network threats and/or other security issues by monitoring computer systems and networks (e.g., for behavioral indicators or suspicious activity). Manual review and investigation of these security alerts is often required to determine if the threats are legitimate and any appropriate actions to take. However, it can take considerable time, effort, and knowledge to perform the manual investigation of just one alert. And since there can be a vast number of alerts generated by these tools in a short amount of time, the manual investigation of every alert is often infeasible and inefficient. Moreover, human review is often flawed and leads to inaccurate or inconclusive results.

[0004] Some organizations have turned to security orchestration, automation, and response (SOAR) platforms to improve the efficiency of their security operations. For example, SOAR

platforms are intended to allow for security alerts to be defined, prioritized, and responded to with reduced need for manual intervention or assistance. However, the reality is that traditional SOAR platforms can be very complex, burdensome, and generate additional workflow for humans. There is often a lot of coding required to automate the tasks involved in investigating an alert, and that code must be constantly updated following changes to investigation procedures or changes to the configurations of computer systems and networks. Accordingly, there exists a need for enabling automatic investigation of security alerts in a manner that is flexible and reduces the burden on humans.

SUMMARY

[0005] For purposes of this summary, certain aspects, advantages, and novel features are described herein. It is to be understood that not necessarily all such advantages may be achieved in accordance with any particular embodiment. Thus, for example, those skilled in the art will recognize the disclosures herein may be embodied or carried out in a manner that achieves one or more advantages taught herein without necessarily achieving other advantages as may be taught or suggested herein.

[0006] All of the embodiments described herein are intended to be within the scope of the present disclosure. These and other embodiments will be readily apparent to those skilled in the art from the following detailed description, having reference to the attached figures. The invention is not intended to be limited to any particular disclosed embodiment or embodiments.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] These and other features, aspects, and advantages of the disclosure are described with reference to drawings of certain embodiments, which are intended to illustrate, but not to limit, the present disclosure. It is to be understood that the accompanying drawings, which are incorporated in and constitute a part of this specification, are for the purpose of illustrating concepts disclosed herein and may not be to scale.

[0008] Figure 1 is a system diagram illustrating an enterprise environment that implements a delegable automated system.

[0009] Figure 2 is a hybrid diagram illustrating components, processes, and data flow within a delegable automated system.

[0010] Figure 3 is a drawing that illustrates an example user interface according to some embodiments.

[0011] Figure 4 is a drawing that illustrates another example user interface according to some embodiments.

[0012] Figure 5 is a flowchart that illustrates an example process that utilizes retrieval augmented generation (RAG) according to some embodiments.

[0013] Figure 6 is a flowchart that illustrates an example process for conducting a threat hunt according to some embodiments.

[0014] Figure 7 is a block diagram that illustrates an example process for investigating an alert according to some embodiments.

[0015] Figure 8 is a block diagram that illustrates an example process for identifying and resolving configuration issues according to some embodiments.

[0016] Figure 9 is a block diagram that illustrates an example process for identifying and ranking issues according to some implementations.

[0017] Figure 10 is a block diagram that illustrates an example process for creating an alert based on an investigation.

[0018] Figure 11 is a block diagram depicting an embodiment of a computer hardware system configured to run software for implementing automated review of security alerts and any systems or methods disclosed herein.

DETAILED DESCRIPTION

[0019] Although several embodiments, examples, and illustrations are disclosed below, it will be understood by those of ordinary skill in the art that the inventions described herein extend beyond the specifically disclosed embodiments, examples, and illustrations and includes other uses of the inventions and obvious modifications and equivalents thereof. Embodiments of the inventions are described with reference to the accompanying figures, wherein like numerals refer to like elements throughout. The terminology used in the description presented herein is not intended to be interpreted in any limited or restrictive manner simply because it is being used in conjunction with a detailed description of certain specific embodiments of the inventions. In addition, embodiments of the inventions can comprise several novel features and no single feature is solely responsible for its desirable attributes or is essential to practicing the inventions herein described.

[0020] It can take considerable time, effort, and knowledge to perform manual review and investigation of an alert to determine if it is indicative of a true threat and, if so, what action to take to address the threat. In some cases, organizations may attempt to automate investigations to at least some degree, but it can take considerable time, effort, and knowledge (e.g., of coding languages, system configurations, data schemas, etc.) to write code for automating tasks involved in investigating alerts. Moreover, code is often specific to a particular platform, particular software, or even specific alerts, presenting a significant challenge as threats, software, and so forth change over time. For example, over time, an organization may add or remove certain applications, security tools, etc., application programming interfaces (APIs) can change, log information can change, and so forth. Adapting to these changes can be a long, difficult process. In some cases, long development times can lead to security gaps, or upgrades may be delayed substantially while new code for security analysis is developed.

[0021] To address these issues, a delegable automated system is described herein that utilizes a human-readable policy and a system configuration ("config"). The human readable policy can specify conditions for investigating alerts, steps for investigating different types of

alerts, and so forth. In some embodiments, the policy, configuration, or both are written in a way that is easy for a human to read and understand (e.g., not in code), although in other embodiments the policy, configuration, or both are written in a more formal manner. In some embodiments, the policy is written in a natural language using a controlled vocabulary, and the policy can be changed and updated over time. In some embodiments, the controlled vocabulary can be updated over time. To automate the investigation of a given alert, the policy and config are used as guides in prompting an AI model (e.g., a large language model (LLM)) to generate an investigation process (e.g., a deterministic investigation process), in the form of code, for example code for querying a data lake or otherwise interacting with a security platform or orchestration environment. In some embodiments, both the process and its outputs are appended to an investigation results document (e.g., a notebook) or used to create an investigation results document, which can be stored for later retrieval. In some embodiments, the results document can be a live document. For example, a user can, in some embodiments, rerun analyses, perform additional analyses, and so forth.

[0022] The alert, the policy, the configuration, the results, or any combination thereof can be used as guides in prompting an AI model (e.g., an LLM) to generate queries, questions, summaries, etc., for example as described herein. The report can recap the investigative steps and their implementation in code; summarize the outcomes of the steps, individually, as a whole, or both; give policy based recommendations driven by the outcomes, which can include system actions, further investigation, routing to others, or any combination thereof; and so forth. Structured elements of the report may also be used in further processes, such as outboard scoring or tagging processes.

[0023] In some embodiments, there may be specific structures or forms to the policy, the config, the alert, and/or the report; and the delegable automated system and its models may be specifically tuned to these forms. In some embodiments, the models may be tuned for common enterprise environments. In some embodiments, the models may be tuned to the mechanism to turn an alert, the config, and the policy into a valid process; and/or the mechanism to turn the results of the process with the alert, the policy, and the config into a

report. In some embodiments, these mechanisms may generate feedback that can be used as training data for the models to improve them. In some embodiments, the approaches herein utilize retrieval augmented generation to provide contextual information, which can help an AI model (e.g., LLM) perform well without retraining even as new threats are discovered, new information about threat actors becomes available, changes are made to a data lake, etc.

[0024] Figure 1 is a system diagram illustrating an enterprise environment 100 that implements a delegable automated system 112.

[0025] For the purpose of facilitating ease of understanding, a singular endpoint 102 and agent 104 is shown in the figure, but it is to be understood that there can be multiple endpoints and agents within the context of an enterprise environment 100. More specifically, there may be a plurality of users (not shown) that use a plurality of endpoints, including an endpoint 102. Some non-limiting examples of endpoints include devices such as smartphones, tablets, desktops, laptops, workstations, servers, and Internet-of-Things devices (e.g., cameras, lighting, refrigerators, security systems, smart speakers, and thermostats).

[0026] In some embodiments, an instance of an agent software application may be installed and executed on one or more of the endpoints, such as the agent 104 running on the endpoint 102. Each agent may be configured to monitor its respective endpoint to detect potential malicious threats (e.g., malware) to the endpoint, the network, and/or the enterprise environment 100. In some embodiments, the agents may detect instances of anomalous/suspicious activity or behavior (e.g., for their respective endpoints or users) and generate alerts associated with those detections. An alert may serve to document and provide information for investigation of a threat that the alert is associated with. In some embodiments, an agent may detect anomalous/suspicious behavior on other endpoints, for example other endpoints that are not running an agent software application. For example, an agent can detect anomalous network activity originating from another endpoint.

[0027] In some embodiments, there is a unified alert inbox 106 that is filled with alerts for different types of potential threats to the enterprise environment 100. For example, one type of threat may be phishing attacks, and one of the alerts may correspond to a particular instance of a suspected phishing attack. Thus, the alerts in the unified alert inbox 106 may collectively document the many instances of potential threats of different types that can be investigated. These alerts can come from various sources, including cybersecurity-related platforms, services, software, tools, and their components. As previously mentioned, the agents running on their respective endpoints may detect potential threats and generate alerts (e.g., endpoint alerts) that are routed to the unified alert inbox 106. The alert inbox 106 can be, for example, a table in a database or data lake. There may also be alerts that originate from third-party sources 120, such as a third-party identity and access management (IAM) service. For example, one of the third-party sources 120 may be a service that authenticates users seeking to access a protected resource from one of the endpoints, and the service may provide alerts on suspicious login behavior that it detects. In some embodiments, alerts from the third-party sources 120 can be stored in the alert inbox 106. In some embodiments, alerts in the alert inbox 106 are be scored and/or ranked based on one or more criteria, such as severity, likelihood that the alert is for a true threat, and so forth. By investigating alerts in the alert inbox 106, determinations can be made about whether the alerts correspond to an actual threat and whether further action is needed, such as taking steps to address a threat.

[0028] In some embodiments, there is a manual review platform 110 that allows a person (e.g., an analyst) to review alerts in the alert inbox 106. In some cases, the analyst may wish to the alerts scoring or other triage techniques or metrics to prioritize which alerts are reviewed. As an example, the analyst may take an alert from the top of the alert inbox 106 and review it (e.g., when alerts are sorted in a ranked order). Based on a quick review, the analyst could determine that the alert reflects a false positive and decide to dispose of the alert immediately. Alternatively, the analyst may decide that the alert could be associated with a real threat and needs further investigation. In some cases, the further investigation may continue to be conducted using the manual review platform 110. For instance, the analyst

may decide to delegate the alert to a junior analyst to review within the manual review platform 110, and the junior analyst can perform an analysis and come back with a recommendation. In some cases, an analyst might escalate an alert for review by a more senior analyst or an analyst with more specialized knowledge related to the alert.

[0029] In some embodiments, there is a delegable automated system 112 that can further investigate the alert. The delegable automated system 112 may be able to investigate the alert, perform an analysis, and/or provide a recommendation. In some embodiments, an analyst may be able to choose to delegate an alert to the delegable automated system 112 for further investigation. For example, the analyst can provide a gesture or select an option within a user interface of the manual review platform 110 to engage the delegable automated system 112 to review a particular alert or multiple alerts.

[0030] In some embodiments, the delegable automated system 112 is configurable (e.g., programmatically, through a user interface, or within the policy 116) to automatically perform investigation of alerts in the alert inbox 106 that meet a certain set of criteria. For example, the delegable automated system 112 can be configured to monitor the alert inbox 106 and automatically investigate alerts for a particular type of event and/or that meet a certain set of criteria (e.g., alerts that are scored/ranked higher than a threshold and/or alerts that are directed to a particular kind of issue (e.g., ransomware, denial of service, phishing).

[0031] In some embodiments, the delegation logic (e.g., determining whether the delegable automated system 112 should automatically review the alert) may be defined in the policy 116. For example, the policy 116 may define the different contexts in which the system 112 should be invoked instead of having the system 112 not take action by default.

[0032] In some embodiments, an alert may either be non-delegable or delegable. A non-delegable alert may require manual investigation (e.g., via the manual review platform 110). For example, the manual review platform 110 may not permit a non-delegable alert to be passed off to the delegable automated system 112 (e.g., that option would not be available). A delegable alert may be manually investigated, but it may additionally or alternatively be delegated to the delegable automated system 112 for further investigation. In some

embodiments, an alert is delegable if the policy 116 contains instructions for how to handle that type of alert. In some cases, the policy 116 can be manually updated for a particular type of alert, and alerts of that type would become delegable to system 112. In some embodiments, the policy 116 can be automatically updated, for example as the capabilities of the delegable automated system improve over time. In some embodiments, updates to the policy are suggested but not implemented until confirmed by a user (e.g., by a senior analyst).

[0033] In some embodiments, there is a contextual bandit 108 that is configured to use contextual information to decide whether to delegate a particular alert to the delegable automated system 112 to investigate or to preserve that alert for manual review (e.g., decide which of the models in an ensemble is appropriate). For instance, the contextual bandit 108 may be aware of past history of human and/or automated system outcomes (e.g., how often did a human look at the automatically generated output and decide it was wrong) when investigating a particular type of alert or under a particular set of circumstances, and so forth. In some cases, the contextual bandit 108 is aware of past outcomes or attributes associated with particular analysts (e.g., person A has domain knowledge applicable to alerts of this specific type). Given its knowledge of past performance, the contextual bandit 108 may be able to allocate a particular alert to the approach that is likely to provide the best outcome. For example, the contextual bandit 108 may delegate the alert to the delegable automated system 112 for review, leave the alert for manual review, or direct the alert for manual review by a particular analyst. In other words, the contextual bandit 108 may be able to learn that in a particular context, the delegable automated system 112 is wrong more often and not as effective as manual investigation, so the alert will not be delegated to the system 112. The contextual bandit 108 may also be able to learn that in other contexts, the delegable automated system 112 is better than a manual investigation (e.g., its investigation is more effective than manual investigation), and so it is better to automatically delegate the alert to the delegable automated system 112 for review.

[0034] In some embodiments, user interaction with the delegable automated system 112 may be via an open chat-based approach (e.g., a chatbot). For instance, there may be a chat

window within a user interface of the manual review platform 110, and the user may be able to converse within the chat window to delegate a particular security alert to the delegable automated system 112 for review. The delegable automated system 112 may be able to provide updates in the chat window as it investigates the alert (e.g., "reviewing the alert," "generating process code," "outputting results to report," "report is available for review," and so forth). In some embodiments, each chat interaction with the delegable automated system 112 is compartmentalized as a separate notebook.

[0035] In some embodiments, there is a system configuration or config 114. The config 114 may specify the settings, arrangement, and availability of components (e.g., programs, applications, services), tools, and data sources available to the enterprise environment 100. In some embodiments, there may be a human-readable policy 116. The policy 116 may specify steps and conditions for investigating different types of alerts, and it can be written in a way that is easy for a human to read and understand (e.g., not in code). For example, the policy 116 can be written in a natural language using a controlled vocabulary, and the policy 116 can be changed and updated over time. More information about the config 114 and the policy 116 is provided in connection with FIG. 2.

[0036] In some embodiments, once an alert has been delegated to the delegable automated system 112 for investigation, the system 112 may generate executable code for a deterministic investigation process based on the alert, the config 114, and the policy 116. More specifically, the config 114 and the policy 116 can be referenced as guides in prompting AI models 118 to generate the executable code, which can be validated and executed to automatically carry out investigation of the alert by performing the steps defined in the policy 116. In some cases, these steps may involve retrieval of data from third-party sources 120 and/or internal data sources (not shown). The results and findings can also be fed into the AI models 118 to generate a report, which can be a structured document that summarizes the investigation, provides results and findings of the investigations, and provides recommendations or suggested actions to implement. In some embodiments, a user may be able to review the generated report (e.g., within the manual review platform 110).

[0037] FIG. 2 is a hybrid diagram illustrating components, processes, and data flow within a delegable automated system according to some embodiments.

[0038] In some embodiments, there may be a unified alert inbox 202 that is filled with alerts for different types of potential threats. The delegable automated system may review an alert, such as alert 204, in the inbox 202. The alert 204 may identify a problem or event (e.g., a potential network threat) and may contain additional information about the problem, such as a diverse set of information associated with the problem or a diverse set of information that can be further investigated. Some of the information in the alert 204 may be gathered directly from the enterprise environment (e.g., the endpoints in the enterprise environment 100 of FIG. 1), but some of the information may also come from third-party sources 218. Accordingly, the alert 204 can be an object with a diverse set of information obtained from many deterministic sources, including third-party sources 218.

[0039] For example, the alert 204 may identify one or more detections that occurred, such as the detection of a certain kind of behavior (e.g., a behavioral indicator) or anomalous activity (e.g., from a user or machine). There can be detection of multiple behavioral indicators or anomalous activities that are associated with an event. In some cases, a particular behavioral indicator may have a known association to a type of network threat (e.g., this behavioral indicator is associated with this type of malware or ransomware). However, the presence of that behavioral indicator and its known association may not be enough information and sufficient cause by itself to convict (e.g., to determine that the behavior indicator reflects a true security incident or threat). In some cases, the detection of anomalous activity may be the output of an anomaly detection system. For example, an endpoint or the user of an endpoint may be doing something unusual that they do not normally do. The alert may also contain additional information about the user, the endpoint, or both, or include an indication that can be used to determine additional information (e.g., a username, machine name, etc.). Accordingly, the alert 204 can include many different things, including the presence of an indicator that is non-convicting and/or an anomalous activity from a user or a machine.

[0040] Anomalous activity can be identified from many different approaches because activity can be considered unusual based on many different standards, and a combination of different standards can also be used. One approach is to identify anomalous activity that is potentially indicative of security issues. For example, a baseline can be established for normal activity and used to identify anomalous activity that deviates from the baseline. The anomalous activity can be evaluated based on a combination of weight of evidence, actual anomalousness, and the overall context surrounding that activity in order to determine whether that activity is interesting or useful. For instance, activity found to be unusual in a certain context (e.g., certain conditions are true – such as the user initially exhibited suspicious login behavior or logged in from an unusual geographic location) may push the activity towards being interesting or useful. In some cases, different instances of anomalous activity can be scored and ranked based on criteria such as the presence of other indicators or the context surrounding that activity. For example, if a user signs in from an unusual location and begins downloading large volumes of information from network shares, this can indicate that an attacker has gained access and is stealing information, although it could also indicate that the user is traveling and is downloading files for offline use. In some embodiments, additional detail is captured which can help distinguish between malicious activity and non-malicious activity. Continuing with the example, if the files being downloaded are files that the user frequently works with, the likelihood that malicious activity is occurring can be lower than the if the files being downloaded are rarely or never used by the user or have no relation to the user's job. In some embodiments, baseline activity can be particularized to an endpoint, a set of endpoints, a user, a set of users, etc. For example, different endpoints and/or users can be expected to have different access patterns, application usage patterns, network usage patterns, etc.

[0041] In some embodiments, alerts in the unified alert inbox 202 may be scored and ranked based on one or more criteria. Some of the alerts can be delegable to the delegable automated system for review (e.g., if the system is configured to handle those types of alerts or potential threats, as may be the case if they are specified in the policy 208). In some cases,

a user may be able to select an individual alert to review (e.g., via the manual review platform 110 shown in FIG. 1) to dispose of it directly. In some cases, the user may be able to manually delegate the alert to the delegable automated system to review. The user may receive an indication (e.g., in a user interface) or an email response to confirm that the alert has been delegated to the delegable automated system to review.

[0042] In some cases, the delegable automated system may be configurable (e.g., programmatically, through a user interface, or within the policy 208) to always be engaged and applied to investigate alerts for a particular type of event or potential threat and/or alerts that meet a certain set of criteria (e.g., alerts that are scored/ranked higher than a threshold and/or alerts that are directed to this particular kind of issue). In other words, the system may review alerts of a particular type of event or potential threat without needing an analyst to manually delegate those alerts to the system. Thus, for those alerts, an analyst will be able to directly review the results produced by the system.

[0043] In some embodiments, once the delegable automated system has picked up a particular alert for review and investigation, the system may provide updates (e.g., via a user interface, by email, or any other electronic communication method). For example, the system may provide updates indicating that the system is working on investigating the alert, adding the analysis results into a document, attaching or providing the document, and so forth. The system may provide a response indicating that review and investigation of the alert is completed and the results are ready for review and/or verification by a human being. An analyst can review the results and take action based on the findings.

[0044] In some embodiments, there may be a config 206. The config 206 (e.g., configuration) may be a file, object, database record(s), or a set of parameters that embodies a system or environment (e.g., enterprise environment 100) In some embodiments, the config 206 specifies the appearance, arrangement, and availability of components (e.g., programs, applications, services), tools, and data sources available to or present in the system or environment. As a more specific example, the config 114 may identify a set of programs, applications, and/or SaaS services (e.g., Okta) available for a particular environment. As

another example, the config 114 may specify the data in the schema or the APIs that can be accessed. As another example, the config 114 may specify various sources or providers, such as an identity provider, a firewall provider, an endpoint provider, and so forth.

[0045] In some embodiments, there may be a policy 208. The policy 208 may be a structured document that specifies the steps that should be taken to investigate the alert 204 (e.g., based on the type of alert 204) and the conditions that would need to be met to make a determination on the alert 204 (e.g., to convict the threat). In other words, the policy 208 may indicate the types of conditions which would convict and the kinds of underlying information those conditions are based on. The policy 208 may also contain recommendations for action or a range of actions to take in certain circumstances.

[0046] In some embodiments, there may be a single policy 208 that effectively serves as one large guide or procedure manual for investigating different kinds of alerts. It may have different subsections in it directed to different kinds of alerts (e.g., types of user information, application logs, system logs, network logs, etc., to consider for a particular type of alert, what steps to perform for a particular type of alert). Accordingly, the type of alert 204 being investigated can be used to determine the appropriate section of the policy 208 that should be applied. In other embodiments, there may be multiple policies that serve as different guides for handling different scenarios, and the type of alert 204 being investigated can be used to determine and select the appropriate policy that should be applied (e.g., policy 208). However, for the purpose of facilitating ease of understanding, the system is generally described herein as implementing a single policy 208.

[0047] In some embodiments, the policy 208 may be written in a natural language format. In some of such embodiments, the policy 208 may be written in a natural language format with a controlled vocabulary. More specifically, there may be words in the vocabulary with protected meanings that the natural language in the policy 208 must adhere to. This ensures that certain key terms are not ambiguous for the system when it interprets the policy 208. For example, certain asset types (e.g., endpoint, machine, process, thread) may have very specific names. These key terms may be provided in advance to the people tasked with drafting and updating

the policy 208 so that they can use the proper terms when writing the policy 208. This may help avoid situations where the person writing into the policy 208 ends up referring to something different from what they intended. However, outside of these key terms, a person would be able to draft the policy 208 in a way that the person finds natural.

[0048] Accordingly, the policy 208 would typically be written by a person or multiple people, and the contents of the policy 208 can be based on many different considerations. For example, a person could base the contents on existing policy (e.g., an existing policy written by the enterprise/company for how to handle different alerts), on settled policy (e.g., investigative steps generally accepted within the industry), on compliance documentation or regulatory compliance (e.g., regulation that specifies a certain kind of investigation be performed or certain kinds of details to be verified), and/or a more formalized version of tribal knowledge.

[0049] In some embodiments, certain policies can be automatically created, for example based on information obtained from knowledge sources such as security document, blog posts, and so forth. In some embodiments, a security vendor can provide policies. In some embodiments, an organization can use the vendor-provided policies, customize the vendor-provided policies, or draft their own policies that override vendor-provided policies.

[0050] The investigation of an alert by the delegable automated system may be predicated on three objects – the alert 204, the config 206, and the policy 208. These three objects may condition the generation of an investigation process 214. If there are multiple policies, the alert 204 would inform which policy to use. Or, if there is a single policy 208, the alert 204 would inform which part of the policy 208 is applicable or needs to be activated. The alert 204 may additionally provide information necessary to perform further investigation of the alert 204. The policy 208 specifies the steps that should be taken to investigate the alert 204 (e.g., based on the type of alert 204) and the conditions that would need to be met to make a determination on the alert 204 (e.g., to convict the threat). Thus, the information provided in the alert 204 may be applicable to and usable in the steps specified by the policy 208. The

config 206 may specify the tools available to perform the steps that are specified by the policy 208.

[0051] As a more specific example of the interaction among these three objects, assume that the alert 204 is associated with a particular type of threat (e.g., a particular type of anomalous behavior exhibited by a user of an endpoint). To further investigate that type of threat, the policy 208 may specify a set of other behavioral indicators that may have been present at the same time the alert 204 was generated and should also be looked at if available. These behavioral indicators may be embodied in or obtained from particular data, and the policy 208 may specify the set of data (e.g., when/where) that should be retrieved for its association with those behavioral indicators. For instance, to investigate an alert 204 that corresponds to a particular type of anomalous behavior exhibited by a user of an endpoint, it may be helpful to review the behavior of that user on other endpoints. In such a situation, the policy 208 could specify that data should be retrieved from all the machines that the user has accessed within a certain time window. Thus, it can be understood how the contents of the alert 204 and the config 206 can interact with the contents of the policy 208 to inform generation of the process 214.

[0052] At the retrieval block 210, the delegable automated system may take the contents of the alert 204, the config 206, and the human-readable policy 208 to combine them into a proper request, prompt, or set of inputs to be provided to the model 212 to generate the process 214. In some embodiments, this may be akin to collation. There may be a structure for the request, prompt, or set of inputs to be provided to the model 212 in order for it to generate the appropriate output (e.g., executable code; a program). In some embodiments, specific elements of the alert 204, the config 206, and/or policy 208 are retrieved and placed a structure for further processing. In other words, the retrieval step may be aware of the contents of the alert 204, the config 206, and/or the policy 208 and can retrieve syntax items that are relevant to implementing the policy 208. For example, the config 206 may specify an identity provider (e.g., Okta) that possesses data needed for investigation of the alert 204, and at the retrieval block 210, the delegable automated system would know how to syntactically

interact with, and retrieve, the appropriate information in the config 206 needed to query that identity provider. More specifically, the delegable automated system may see that the policy 208 requires accessing data from the identity provider and the config 206 specifies the identity provider to be Okta, and then retrieve from config 206 the Okta API information needed to access that data.

[0053] The model 212 may generate a deterministic investigation process 214 for carrying out investigation of the alert 204. The process 214 may be in the form of code. In some embodiments, the process 214 may be code for a specific orchestration environment. (e.g., Torque). In some embodiments, the process 214 may be code in a more general programming language (e.g., Python). The code may include executable instructions for obtaining information (e.g., accessing databases; accessing data lakes or entire sets of databases; accessing APIs for various third-party systems and services, including issue tracking systems such as Jira or identity management systems such as Okta). The code may include executable instructions for analyzing that information (e.g., processing data, subjecting the data to various conditions, applying calculations or models, performing anomaly detection or statistical analysis, and so forth). The code may also include executable instructions for other tasks beyond data retrieval and analysis, such as establishing an issue in an issue tracking system e.g., an API call to Jira). The code may include executable instructions for generating an output (e.g., making a determination based on the analysis of the data, determining recommendations or suggestions based on the determination, and so forth). In some embodiments, the code can be executed, and the generated outputs can be stored for further use, such as to generate a report 222. In some embodiments, the generated outputs can be stored as objects and associated with the code used to generate the outputs.

[0054] The model 212 can include any artificial intelligence (AI) model capable of generating the process 214. In some embodiments, the model 212 can include a large language model (LLM). And although model 212 is referred to in the singular, model 212 may actually be a plurality of models capable of generating the process 214. For instance, model 212 may be an ensemble of models. In some embodiments, the model 212 may be any suitable model or

ensemble of models that can generate the process 214 by taking instructions written in natural language from the policy 208 and turning it into code.

[0055] Although not depicted in the figure, in some embodiments, the generation of the process 214 may involve performing validation on the code output by the model 212. This validation may be automated, and it can involve checking the output of the model 212 for certain criteria or conditions to be met. If those conditions are met, then the code may be validated deterministically (e.g., to make sure that it is valid Python code, all the API calls are valid, and that the overall series of steps taken are valid). If everything checks out, then the finalized code may become the process 214.

[0056] In some embodiments, the model 212 may be capable of making corrections to code (e.g., when provided with code and annotations of changes to be made), and the generation of the process 214 and the validation on the code output by the model 212 may be an iterative or looping process. For example, the model 212 may be prompted to output code that should meet certain criteria or conditions (e.g., it should be under a certain length). If those conditions are not met, then the model's output can be annotated to identify corrections to be made (e.g., this code is too long) and fed back into the model 212 to update and make corrections. Once the model 212 outputs code that meets the conditions, then the code may be validated deterministically. If there are issues identified in the code during validation, annotations can be made on the code and fed back with the code into the model 212 to update and make corrections. This may continue until the model 212 produces code that appears valid, and that finalized code may become the process 214 or part of the process 214. Thus, going from the retrieval of information at block 210 to the generation of the process 214 may involve multiple steps, and it may include pushing inputs through the model 212 multiple times (e.g., two or three times, or conceivably more) in order to obtain output code that appears valid.

[0057] In some embodiments, the validation of code may involve inputting the code into one or more deterministic validators that are configured to review the code, identify errors, and generate error descriptions and annotations for correcting the code. For instance, there may

be a validator that reviews code to identify syntax errors (e.g., this set of brackets was not closed), there may be a validator that can identify issues with the libraries or APIs referenced in the code (e.g., this method uses a deprecated library which needs to be removed and replaced with this particular library), and so forth. The code output by model 212 can be reviewed by such validators to annotate the code with error corrections, and code and annotations may be fed back into the model 212, which may be capable of modifying the code based on the provided annotations to produce corrected code for revalidation.

[0058] In general, since the policy 208 was used to generate the code in the process 214 and that code has been determined to be valid, the code is very likely to be an embodiment of the policy 208. However, this is not an absolute. Even though the finalized code in the process 214 should be valid, there may not be a guarantee that the code possesses the same and correct intent as that of the instructions specified in the policy 208. For instance, it could be possible that the code is missing some of the details specified in the policy 208. This is because the validity of the code may not be strongly associated with the code's intent. For example, an LLM or other model may generate code that is correct (e.g., executable) but that does not fully or accurately implement the relevant portions of the policy 208. In some embodiments, incorrect intents can be mitigated by, for example, revising the policy 208 to address ambiguities.

[0059] In some embodiments, once a valid process 214 is obtained, it can be immediately executed to perform further investigation of the alert 204. More specifically, the generated process 214 may be configured so that it can be executed to automatically perform further investigation of the alert 204 based on the steps and tasks laid out in the appropriate section of the policy 208. For instance, the process 214 may cause the collection of data as inputs, the performance of analysis of that data, and the generation of outputs (e.g., determinations, recommended actions) and enrichments. These outputs can be saved with the process 214 to aid in generation of a report 222. To summarize, at retrieval block 210, all the necessary information from the alert 204, the config 206, and the policy 208 is collected and provided to the model 212 in a specific format so that the model 212 can generate a deterministic

process 214 that can be validated. The process 214 then can be executed to generate outputs that can then be used to make decisions and determinations as specified by the policy 208.

[0060] At the retrieval block 216, the delegable automated system may take the process 214, the outcomes of the steps in the process 214 (e.g., the outputs generated by the process 214 code), the original information contained in the alert 204, any additional information retrieved during execution of the code or from third-party sources 218, information from config 206, information from policy 208, and provide it to a model 220. The model 220 may be any artificial intelligence (AI) model capable of combining that information and generating a report 222. In some embodiments, the model 220 may be a large language model (LLM). And although model 220 is referred to in the singular, model 220 may actually be a plurality of models capable of generating the report 222. For instance, model 220 may be an ensemble of models. In some embodiments, the model 220 may be any suitable model that can generate a report 222 based on the process 214, the outcomes of the steps in the process 214 (e.g., the outputs generated by the process 214 code), any recommendations or suggested actions, and any context-providing information (e.g., the information contained in the alert 204, config 206, or policy 208, any additional information retrieved during execution of the code or from third-party sources 218, etc.). In some embodiments, the model 212 and the model 220 are the same model or include a same model.

[0061] The report 222 serves as an investigation results document and can be stored for later retrieval. In some embodiments, the report 222 may combine the findings of the process 214 and the policy 208 elements to generate recommendations. In some embodiments, the report 222 may describe the investigation of the alert 204, such as the information that was reviewed, the results of any analysis, the recommended judgment of the alert 204, the recommended action to take for remediation (e.g., based on the policy 208), and so forth. In some embodiments, the report 222 may include a range of actions that might be taken in the specific circumstances. In some embodiments, the report 222 may be understood in terms of the policy 208 and a user reviewing the report 222 would be able to come to their own conclusions about the alert 204 without having to ever write code.

[0062] In some embodiments, the report 222 may be a structured document. As a structured document, there may be elements of the report 222 that are free text. In some embodiments, the report 222 can include further actions that can be performed. For example, the report 222 can include embedded actions that have not yet been executed when an analyst views the report 222. For example, an analyst viewing a report may wish to make further inquiries and may do so by clicking or otherwise selecting an embedded action. The results of executing the embedded action can be populated into the report 222. In some embodiments, the report 222 is rendered and viewable in a user interface. In some embodiments, once rendered in a user interface, an analyst viewing the report can interact with elements of the report 222 to trigger further actions.

[0063] Once the report 222 has been produced, the delegable automated system may perform further actions or processes based on the report 222. An example of a further process would include scoring and/or tagging, as depicted at block 224.

[0064] In some embodiments, the delegable automated system may update the score associated with the alert 204 at block 224. More specifically, the delegable automated system may take certain elements of its output and findings from investigating the alert 204 (e.g., in the report 222 generated for the alert 204) and use it to rescore the alert 204. For instance, in some cases, if the delegable automated system concludes its investigation without being able to make a final disposition for the alert 204 (e.g., the system cannot make a determination with at least a threshold confidence level) then the system may change the risk ranking or score associated with the alert 204. The system may make this change based on the additional information gathered or analysis/judgments made during the investigation. The rescored alert 204 can then be provided back to a person to make the final disposition. The system may make this change based on the additional information gathered or analysis/judgments made during the investigation.

[0065] As a specific example of this, imagine the scenario that the system executes and follows the process 214 generated for the alert 204 based on the config 206 and the policy 208. Based on the evidence available, the alert 204 appears to be benign but it cannot be

conclusively classified as such and should be left for an actual person to decide on. The person may be able to gather much more evidence to weigh, so it is desirable for the person to make the final decision. In some embodiments, the delegable automated system may facilitate a decision on whether to perform the recommendations or suggestions in the report 222 on an automated basis (e.g., actions for remediating or mitigating the threat).

[0066] For the purpose of facilitating ease of understanding, a very specific example of the delegable automated system investigating an alert is now described. Consider that an identity alert associated with anomalous access by a user is generated and received from an identity management service, such as Okta. Also consider that, for this type of alert, if we know a number of machines that have been accessed by the user within the last 48 hours and there are certain indicators present, then that is a strong indication that this is an actual malicious user access. The delegable automated system is tasked with reviewing this alert, either because an analyst has manually delegated the alert to the system or because the system is configured to automatically review that type of alert.

[0067] The policy may specify that this type of alert (e.g., identity alert associated with anomalous access by a user) may warrant further investigation if there are certain indicators present or if user has taken unexpected or unusual actions. The policy may specify steps for investigation this type of alert, such as determining which machines and assets the user has logged into over the last 48 hours and reviewing the pattern of access of these machines and assets by the user over the last 48 hours. The policy can specify investigating for the presence of one or more indicators. In other words, the policy may specify to look at the assets (e.g., monitoring data for the assets) that the user has accessed in the last 48 hours and apply certain conditions or checks to the assets, such as to check for the presence of access from countries which are on a restricted list (e.g., a list of countries defined in the policy), access for more than one location or asset within an hour, and so forth. These conditions are not expressed in a programming language; the conditions can be enumerated in a natural language, which may be constrained with a controlled vocabulary. The policy may also specify that if there is a number of machines that have been touched by this user within the

last 48 hours and any of the identified indicators are present, then that is a strong indication that this is an actual malicious user access and action should be taken in a specified way.

[0068] From these statements, it can be understood that to determine which assets the user has touched, the system will have to query an identity management service or data source that has knowledge of all the assets that the user has touched (and the exact service or data source can be specified in the config). For example, the config may specify that the identity management service in use is Okta and provides additional information on how information about a user can be retrieved from Okta (e.g., via an API with a particular API key). In general, the config may specify additional details for how the instructions provided in the policy can be implemented. In some embodiments, such information may be stored in a data lake, such that queries can be executed against a data lake rather than pulling information from endpoints or accessing logs on third party services.

[0069] The system can retrieve all of this relevant information from the alert, the policy, and the config, and then provide it in a specific format to a model that is configured to generate a process (e.g., executable code) for investigating the alert based on the instructions in the policy. In this instance, the retrieval process is configured to combine the relevant contents of the policy, the relevant contents of the config, and syntactic information about how to use the Okta API (and/or any additional useful information), into a format that can be fed into the model, so that the model generates a process that executes the instructions in the policy using the provided syntax.

[0070] This process can be dynamically generated and conditioned based on the contents of the alert, the policy, and config. For example, in this instance, this process would include code for querying Okta to retrieve all the access information for the user in the last 48 hours. However, if a different identity management service or data source were implemented instead in the environment, it would be specified in the config and the generated process would involve querying that instead. In this instance, the process would also include code to process and analyze the retrieved access information to check for the presence of the other indicators or the conditions specified in the policy, such as if the user accessed assets from

countries on the restricted list or accessed more than one location or asset within an hour. If the process is valid (e.g., the code is valid), then it can be executed to carry out the investigation of the alert. Any results and findings from executing the process can be saved and can include any recommendations or suggested actions based on those findings.

[0071] The system can generate a report by retrieving and enriching all the relevant information contained within the alert, the policy, the generated process, and the results/findings from executing the process. This report may state the information the system looked at, the steps performed by the system, the results and findings of the analysis, and any recommendations or suggested actions for disposition. The report is intended for a person to review, and it can contain context and be structured in a manner such that the reviewer can easily understand the alert, the investigation that was performed, the findings from the investigation, and the recommended actions to take. In some embodiments, the reviewer can take recommended actions during review of the report, such as via a user interface element (e.g., clicking a button to execute code embedded in the report that carries out a recommended action).

[0072] In certain cases, the system may be able to make conclusive determinations (e.g., determinations with a confidence above a threshold level), and it may be apparent to the reviewer of the report to apply the recommended action. In some cases, the system can be configured to automatically execute a recommended action without human approval. However, this may not always be the case, and there may be ambiguity present that prevents the system from making a conclusive determination. However, the system may be generally aware of those situations and be able to flag the report for further review so that the policy can be updated to better handle those situations.

[0073] For example, consider the instance in which the policy specifies to check for the presence of access from countries on a restricted list that includes “Korea” (as opposed to specifying North Korea or South Korea). Or alternatively, the policy does specify which Korea but the retrieved access information for the user only indicates “Korea.” The system and the model generating the process may know that North Korea and South Korea are more specific

than Korea (e.g., due to meaning of words), and this ambiguity may result in the report being flagged for additional review due to the ambiguity or the difficulty in making a conclusive determination.

[0074] Accordingly, there are many different novel concepts disclosed herein, and the delegable automated system and its various implementations described herein may have numerous benefits, advantages, improvements over systems or approaches configured for collecting data, generating and reviewing security alerts, and responding to network threats.

[0075] The delegable automated system and its various implementations described herein may reduce the amount of work generated for human beings. In general, an enterprise or organization may only have a small number of people tasked with reviewing security alerts and network threats. There can be a large number of alerts, which translates into a large amount of work to be done to review them all. Furthermore, it can be difficult to determine where to start and which alerts should receive priority for reviewing. It can also be difficult to determine the appropriate actions to take to review an alert or remediate a threat. Thus, it can be very time consuming for analysts to properly review all the alerts, and often beyond the capabilities of analysts. The delegable automated system and its various implementations described herein include the automated performance of various tasks improve the security of endpoints and networks by enabling faster and/or automated analysis of alerts. In some embodiments, the approaches described herein can include ranking/scoring of the alerts (e.g., in the unified alert inbox), reviewing the alerts, and then generating recommendations. In some embodiments, confidence scores or other metrics can be included in a report, alert summary, etc., which can help analysts identify which alerts are most likely to indicate a true security threat.

[0076] The delegable automated system and its various implementations described herein may provide consistency of action. For a particular type of alert and threat, the policy 208 is consistently interpreted and embodied into action the same way each time. Instead of having differences in interpretation by multiple human beings (e.g., associated with differences in steps taken in review/investigation of the alert), there is a single interpretation that is

consistently applied. When it comes to human beings, even a single individual can make different interpretations in similar situations; the individual can forget things or misapply the accepted policy/guidelines for reviewing the alert. In comparison, the delegable automated system can apply its policy 208 much more consistently, which can also lead to much better results overall if the delegable automated system can apply its policy 208 better than analysts.

[0077] The delegable automated system and its various implementations described herein may allow analysts to focus on policy and the overall approach for reviewing and remediating a type of alert and threat (e.g., writing and updating the contents of policy 208) instead of execution and the actual steps of reviewing an alert. Analysts can focus more on the actions that should be taken in different circumstances by classifying the different types of threats and circumstances that can arise and specifying the policy and actions for handling each type (e.g., within the policy 208). Analysts can focus on writing the policy 208 by abstractly specifying the core process of the investigation, and the alert 204 and the config 206 can provide the information used for implementing the policy 208. Advantageously, because the policy 208 can be in a natural language format, analysts are able to define the actions that should be taken in each circumstance without needing to have extensive security and coding ability/experience.

[0078] Advantageously, analysts do not need knowledge of specific implementation details, such as the various APIs used within the enterprise environment, query syntax, and so forth. For example, a third-party identity management service may have an API for requesting data, and traditional review of an alert may involve a security analyst making a request via this API. This would normally require a large amount of knowledge (which can change over time as APIs are updated) of the underlying API and how to use the API to perform the investigation efficiently. More specifically, it requires knowledge of the correct API calls to make, how to structure those API calls to retrieve data in the format needed to properly review the alert, and how to structure those API calls to limit the retrieved data to just the desired data. In contrast, the delegable automated system can make these details part of the policy 208, the config

206, and/or the retrieval/collation mechanism (e.g., at retrieval block 210). For instance, the retrieval/collation mechanism can be provided access to the API documentation that explains the various API calls and formats, and the relevant information from that documentation can be provided to the model 212 to generate the process 214.

[0079] Furthermore, the delegable automated system and its various implementations described herein may provide fluidity in various ways. More specifically, it may provide the ability for an enterprise or organization to quickly and easily (e.g., without requiring a lot of manual work to be performed) change how they investigate and remediate a type of network threat, since the desired approach or aspects the enterprise environment can change frequently over time. For example, if a tool or aspect of the enterprise environment changes (e.g., there is a change in the third-party identity management service), a change can just be made in the config 206. The retrieval/collation mechanism can also be provided access to the API documentation if necessary. No changes would have to be made to the policy 208 to enable the model 212 to continue generating code for reviewing alerts. Alternatively, to change the approach for investigating a type of alert, a change can be made in the policy 208. No changes would have to be made to the config 206 to enable the model 212 to continue generating code for reviewing alerts. This allows changes to be quickly implemented without having to inform people of changes and having them constantly update code, settings, hookups, etc.

[0080] Accordingly, the delegable automated system described herein may have numerous advantages over traditional systems in cybersecurity that are configured for the tasks of collecting data, reviewing security alerts, and responding to network threats. An example of such a traditional system is security orchestration, automation, and response (SOAR) software and systems. Although the aim of SOAR systems is automation with reduced human assistance, the reality of SOAR systems is that they generate additional workflow for humans. Typically, SOAR systems lack the same level of abstraction and will require their users to directly write code for investigating an alert. The introduction of new conditions or changes requires users to write new code or make direct changes in code. This is a rigid approach that

requires a deep understanding and extensive knowledge of the components and overall configuration of the enterprise environment, how those components are hooked up, the schemas of the information need to perform an investigation, the steps involved in the investigation, and the coding languages in which everything is expressed. There is a lot of ongoing investment to stay on top of all this knowledge and continually write code to investigate alerts based on that knowledge. Furthermore, any changes typically require a large amount of manual work to implement. To put it simply, the continued use of SOAR systems requires a large amount of continual maintenance performed by humans. In some cases, this additional work may lead to people not using the system, which reduces their efficacy. In contrast, the delegable automated system described herein allows code to be dynamically generated even when there are constant changes to investigation procedure or the enterprise environment, thereby enabling automatic investigation of security alerts without the pain and maintenance cost associated with typical SOAR systems.

Guided Investigation

[0081] As described herein, analysts are often under-resourced, which can make investigating potential security issues difficult. In some cases, investigations can be delegated to junior analysts who may lack a deep body of experience and thus may struggle to perform thorough investigations, potentially taking too long and/or missing some important steps in an investigative process. Even experienced analysts can make oversights, errors, and so forth. Gaps, errors, inconsistencies, and so forth in investigations (or lack of investigation at all in some cases) can result in significant security issues. Advantageously, the approaches described herein can improve computer system security.

[0082] In some cases, analysts can operate in a reactive manner. That is, an investigation can be conducted after a potential issue has been identified. In some cases, investigations can be proactive. Threat hunting can be used to proactively search for undetected security threats. For example, while endpoint detection software, antivirus software, and so forth can identify many threats, in some cases a threat may escape detection by such tools. Persistent threats can remain undetected for a significant period of time, potentially compromising large

amounts of data, disrupting operations, and so forth. In some embodiments, investigations can be carried by looking for indicators of compromise (IOC), evidence of tactics, techniques, and procedures (TTP), and so forth. However, performing investigations typically involves large amounts of data, complex queries, and so forth, making such work difficult or impossible for junior analysts. Even experienced analysts may struggle to perform a thorough investigation, as computer systems, network infrastructure, software, and so forth can be complex and can change quickly.

[0083] IOCs can include, for example, network traffic anomalies, unusual sign-in attempts, privilege escalation attempts, system configuration changes, unexpected software installation and/or updates, repeated requests for the same file or data, unusual DNS requests, and so forth.

[0084] TTPs can play an important role in addressing cybersecurity issues. Tactics refer to high-level descriptions of a threat actor's behavior. Techniques provide context for the tactics, and procedures describe the specific steps or activities for carrying out attacks. In some embodiments, TTPs are accessed from a third party source, such as MITRE® ATT&CK®.

[0085] Tactics are the overarching strategies and goals behind an attack. Tactics describe, at a high level, overall strategies and goals of an attack. Understanding tactics can be important because it can provide insight in the threat actors, which can in some cases be identified by the use of specific tactics.

[0086] Techniques describe the methods that threat actors use to carry out an attack. Techniques can be used to gain initial access, move laterally within a compromised environment, steal data, and so forth. Procedures describe particular sequences of actions that make up an attack. Procedures include the specific tools used by a threat actor.

[0087] In some embodiments, a machine learning model (e.g., a large language model (LLM)) can be used to guide an analyst through an investigation. For example, in some embodiments, an analyst can select a type of analysis to conduct and can be guided through the analysis. For example, in some embodiments, an analyst can select to conduct a hunt

based on threat actor, indicators of compromise (IOC), tactics, techniques, and procedures (TTP), entity, malware, campaign, asset, alert, and/or anomaly. It will be appreciated that these are only examples, and a system can have more, fewer, and/or different hunting types. Hunts can be manual, partially automated, or fully automated. For example, in a manual hunt, a user can supply questions (e.g., natural language questions) to the model to conduct an investigation. In a partially automated hunt, the model can suggest questions to the user, for example based on previous questions, the results of previous questions, and so forth. In a fully automated approach, the model can automatically generate and/or select questions, for example based on a pre-defined set of questions and/or by automatically generating questions based on previous questions and/or results of previous questions.

[0088] In some embodiments, when an analyst selects a hunting type, the system can create a new notebook. The notebook can serve as a log of queries, results, and so forth. In some embodiments, when a user selects a hunting type, a first hunting question can be automatically executed or suggested to the user.

[0089] In some embodiments, the system can provide the analyst with suggested questions. The suggested questions can help to guide the analyst through the hunt. Suggested questions can be based on various factors, such as questions that have already been asked, the results of queries executed in response to questions, and so forth.

[0090] In some implementations an analyst can select a threat actor hunt type. In some embodiments, the system can execute a default question related to the hunt, such as “What threat actors have the most associated IOCs and TTPs present in my environment?” The system can suggest follow up questions related to the hunt, such as:

- What specific IOCs are present for the top threat actor?
- What TTPs are present for the top threat actor?
- What machines have the most IOCs and TTPs?
- Which TTPs are most present for the top threat actor?

- Which IOCs are most present for the top threat actor?

[0091] In some embodiments, an analyst can select a TTP hunt type. In some embodiments, the system can execute a default question related to the hunt, such as, "What TTPs occur most in my environment?" The system can suggest follow up questions related to the hunt, such as

- What TTPs are occurring least in my environment?
- What machines are involved with the most frequent TTP?
- What users are involved with the most frequent TTP?

[0092] In some embodiments, an analyst can select an IOC hunt type. In some embodiments, the system can execute a default question related to the hunt, such as, "What IOCs are most common in my environment?" In some embodiments, the system can suggest follow up questions related to the hunt, such as:

- What IOCs are least common in my environment?
- What machines are involved with the most common IOCs?
- What users are involved with the most common IOCs?

[0093] In some embodiments, an analyst can select an asset hunt type. In some embodiments, the system can execute a default question related to the hunt, such as, "What machines have the most indicators on them, broken out by indicator type." In some embodiments, the system can suggest follow up questions related to the hunt, such as:

- For the top occurring machine, what specific IOCs/threat intelligence indicators are on it?
- For the top occurring machine, what specific TTPs?
- For the top occurring machine, what business intelligence is on it?
- For the top occurring machine, what specific static indicators are on it?

[0094] In some embodiments, an analyst can select a malware hunt type. In some embodiments, the system can execute a default question related to the hunt, such as "What malware families are most occurring in my environment?" In some embodiments, the system can suggest follow up questions related to the hunt, such as:

- What malware families are least occurring in my environment?
- What machines are involved with the most occurring malware families?
- What users are involved with the most occurring malware families?

[0095] In some embodiments, an analyst can select a campaign hunt type. In some embodiments, the system can execute a default question related to the hunt, such as "What campaigns have the most associated IOCs and TTPs present in my environment?" In some embodiments, the system can suggest follow up questions related to the hunt, such as:

- What specific IOCs are present for the top campaign?
- What TTPs are present for the top campaign?
- What machines have the most IOCs and TTPs for the top campaign?
- Which TTPs are most present for the top campaign?
- Which IOCs are most present for the top campaign?

[0096] In some embodiments, an analyst can select an anomaly hunt type. In some embodiments, the system can execute a default question related to the hunt, such as, "What log sources are most common in my environment?" In some embodiments, the system can suggest follow up questions related to the hunt, such as:

- For the top log source, are there unusual events?
- For the least occurring log source, are there any unusual events?

[0097] As described above, in some embodiments, a system can be configured to automatically execute a first query based on a selected threat hunting type. However, this is not necessarily the case. In some embodiments, a system can be configured to generate one

or more suggested questions for a user based on a selected threat hunting type, and the user can select from a plurality of questions to begin an investigation. The selected hunting type can provide context for generating suggested questions.

[0098] In some embodiments, questions may not be context aware. For example, questions may not consider the environment against which they are being asked, current threats, etc. In some embodiments, questions can be context aware. For example, in some embodiments, questions can be determined at least in part based on current threat information, trends across a customer base (e.g., questions that are commonly being asked by customers), and so forth. For example, if there are recent reports of activity by a threat actor, the system may ingest this information and suggest a question related to determining whether or not the threat actor has conducted any suspicious activity against the environment. As another example, if there are reports of a new malware, the system can suggest questions related to the malware.

[0099] As the user proceeds through an investigation, the system can suggest questions, for example based on questions already asked by the user, the responses to questions, questions asked by others conducting similar investigations, questions that have been predetermined to be probative, and so forth.

[0100] In some embodiments, the system can generate suggested questions based on the results of previous questions. For example, if a user asks for a list of IOCs, the system can return a table that includes asset names and IOCs. The system may then suggest, for example, asking further questions about an asset with a high number of IOCs.

[0101] As described herein, the suggested questions can adapt based on answers to questions, previously asked questions, and so forth, thereby enabling a user, even a junior analyst, to perform thorough analysis with relative ease. As an example, consider a suspicious user who belongs to a particular user group. The system may recommend questions such as “what systems has the user logged in to in the last week?” or “what files has the user accessed?” Subsequent questions could be, for example, “How does the user’s access history compare to other users in the same group.” A junior analyst may not think to

ask such a question, but doing so can provide valuable insight as it can show whether the user's access patterns are truly deviating from normal behavior or if the user is behaving like other users in the same group. It can also help to identify specific differences between the user and other users in the group.

[0102] In some embodiments, the system can generate a user interface that allows the user to select from among a plurality of starting points. In some embodiments, the user interface can include an input box that allows a user to submit a natural language question. In either case, the system can be configured to generate a query against a data lake based on the received question. In some embodiments, the system can use an LLM to generate a query based on a question selected by a user and/or input by the user.

[0103] In some embodiments, the system can provide the question to the LLM and can additionally provide information about a database or data lake schema, query language syntax, and so forth to the LLM, such that the LLM can use the question and the syntax and schema information to create a well-formed query. For example, in some embodiments, the system can use retrieval augmented generation to provide information about the data lake to a mode.

[0104] In some embodiments, the LLM can rely on information from a knowledge base when generating a query. For example, a knowledge base may contain information that relates certain threat actors, malware, etc., to certain TTPs and/or IOCs. As one example, if a user asks a question such as, "Am I being targeted by Happy Hacking Co," the system can pull information from the knowledge base such as IP addresses, domain names, etc., that are typically associated with Happy Hacking Co. As another example, if a user wants to know if their environment is being targeted by a specific malware, the LLM can rely on knowledge base information that describes the IOCs that are expected to be present if the environment is being targeted by the malware. As one more example, if a user asks if their environment is being targeted by actors from a rogue nation, the LLM can rely on information from the knowledge base that indicates, for example, threat actors associated with the rogue nation, TTPs used by actors in the rogue nation, and so forth. Thus, a user can ask general questions

without a need for deep technical knowledge or knowledge of specific malware, threat actors, and so forth, and the system can use the knowledge base to generate queries that can be used to assist the user in their hunt.

[0105] In some embodiments, the knowledge base can be a vector database that contains internal information, external information, or both. Internal information can include, for example, documentation, support materials, blog posts, and so forth. External information can include, for example, threat intelligence feeds, data from attack frameworks (e.g., MITRE), and so forth. By pulling information from the knowledge base, the system can better understand a user's request. For example, if a user asks, “Am I being attacked by Apt 729?,” the system can determine that the user is asking about a particular threat actor (Apt 729). The knowledge base can contain information about what the threat actor looks like (e.g., what TTPs they use), IP addresses associated with the threat actor, domain names associated with the threat actor, and so forth.

[0106] The information from the knowledge base can be used in retrieval-augmented generation. Retrieval-augmented generation (RAG) can be used to optimize the output of an LLM. In RAG, an LLM references an authoritative source (e.g., the knowledge base) prior to generating responses, such as a query against a data lake. RAG can help to mitigate or eliminate problems associated with LLMs, such as hallucinations, out-of-date information, inaccurate responses, and so forth. With RAG, even though the LLM may not have been trained on the latest data and/or may be trained using a wide variety of data, at least some of which may be unrelated to the task at hand, relevant, reliable responses can be obtained.

[0107] In some embodiments, third-party LLMs can be used. In some embodiments, the system can use a custom LLM. In some embodiments, multiple LLMs can be used, and the particular LLM to use for a particular task can be determined based on the task to be performed. For example, one LLM may be better at summarizing information, while another may be better suited to generating queries in a particular language. In some embodiments, cost can be considered when selecting an LLM to use.

[0108] In some embodiments, users can be provided with raw query output. In some embodiments, it can be desirable to provide the user with a summary of the output. For example, query results can be provided to an LLM and the LLM can summarize the results, for example identifying the most affected systems, summarizing how many IOCs were found, explaining TTPs that were uncovered, and so forth.

[0109] In some embodiments, an LLM can be used to determine one or more descriptive statistics, such as total number of affected assets, maximum number of TTPs/IOCs on an asset, minimum number of TTPs/IOCs on an asset, average number of TTPs/IOCs, median number of TTPs/IOCs, outliers, and so forth.

[0110] Explainability can be key to ensuring that users trust the results they receive from the system. Accordingly, in some embodiments, the system can provide the user with information such as the exact query that was executed. In some embodiments, the system can be configured to provide sources for information. For example, if a summary relies on data from a third party, documentation, a blog post, etc., the system can provide a link or other citation indicating where the information came from.

[0111] Maintaining accurate records of investigations can be important, for example to identify potential gaps in investigations, to understand what was done during an investigation, to perform repeat investigations, and so forth. In some embodiments, a user can create a new notebook, or the system can create a new notebook when a user begins a new investigation. The notebook can be a log of prompts, queries, results, and so forth that allows the user, the user's peers, the user's manager, and so forth to review exactly what was done in the course of the investigation.

[0112] In some embodiments, notebooks can be saved, shared, searched, and so forth. In some embodiments, the system can provide personal notebooks and shared notebooks. Personal notebooks can be notebooks associated with a particular user. Shared notebooks can be notebooks that are shared with others (e.g., with a team or manager). Different users may have different permissions for different notebooks. For example, some users may only be

able to read a notebook, while other users may be able to ask additional questions, rerun analysis, and so forth.

[0113] In some embodiments, users (or other users) can rerun analysis. This can be significant as there can be more information available over time. For example, if an attack is ongoing, it can be significant to rerun analysis to determine additional affected systems, the current (or recent) extent of an attack, and so forth. Rerunning an analysis can, additionally or alternatively, help determine the success of mitigation efforts. By maintaining a record of previous analyses, it can be easy to rerun the same analysis, thereby ensuring that the updated results are suitable for comparing with previous investigation results.

[0114] In some embodiments, the user can advantageously provide an intent with an initial question. For example, if the user selects a “threat actor” hunt type, the system can use this information to determine that additional question suggestions should be based around the idea of investigating for evidence of a threat actor.

[0115] In some embodiments, the system can determine that certain questions are of high importance for an investigation. In some embodiments, the system can repeatedly suggest a high importance question to a user until the user asks the question. In some embodiments, the system can be configured to automatically run certain questions that are of high importance.

[0116] In some embodiments, an investigation can follow a tree-like structure. However, it will be appreciated that the user is not simply following a pre-determined decision tree. Rather, the user can ask questions in an order of their choosing, and the suggested questions are not strictly defined by a decision tree, but rather suggested based on the results of the investigation so far, the questions already asked, the type of investigation, and so forth.

[0117] In some embodiments, a user can pick out a specific asset for investigation. In some embodiments, the system can suggest specific assets for further investigation. For example, the system can suggest assets for further investigation based on which assets have the most IOCs, which assets have the most recent IOCs, and so forth. In some embodiments, the

system can recommend assets for further investigation based on criticality, function, and so forth. For example, the system may have information indicating that a particular asset is a critical asset, is a domain controller, and so forth, and can suggest further investigation of such assets. In some cases, an organization may manually identify critical assets, domain controllers, and so forth. However, such manual identification can be laborious, can become outdated or incomplete, and so forth. Thus, in some embodiments, the system can automatically determine criticality, asset type, and so forth, for example based on services running on an asset, machine name, and so forth. For example, a machine name that contains a letter combination such as “ad” or “ldap” or “dc” can likely be an Active Directory server, LDAP server, or domain controller, respectively. Similarly, a name that contains a term such as “laptop” or “desktop” can indicate that the endpoint is a machine used directly by end users, rather than, for example, a file or mail server.

[0118] In some embodiments, the user can add tags to the notebook. The tags can indicate information such as the conclusion, reason for the investigation, and so forth. In some embodiments, the system can be configured to automatically generate tags, for example based on the contents of the notebook. Tags can make for relatively easy searching of notebooks. For example, if an organization has previously investigated for particular threat actors, IOCs, etc., an analyst may want to find a previous investigation, for example to rerun the investigation, add on to the investigation (e.g., by asking additional questions), or simply to use as a reference for determining which questions to ask in a new investigation.

[0119] The size of the notebook can be significant. For example, LLMs can be trained and/or can use the contents of notebooks when using RAG. However, LLMs typically have a limited attention window. Thus, it can be important to ensure that notebooks are small enough to have a significant portion (e.g., all or most) of the notebook’s contents fit within the attention window, thereby ensuring that context is not lost. Thus, for example, in some implementations, only subsets of results are stored within the notebook, or only links to results are included in the notebook. For example, a large number of results of a data lake query may be of limited value and may diminish performance of an LLM. As an example, if a

question asks for the endpoints that are most impacted by a particular technique, the results of a query can be a ranked listing of endpoints, for example sorted in descending order of impact. Endpoints outside of the top few results (e.g., the top 1 result, top 5 results, top 10 results, top 100 results, etc.) may be of little or no significance. The number of relevant results can vary and can depend on, for example, the severity of an attack, the number of impacted systems, and so forth.

[0120] In some embodiments, the system can suggest follow on actions in addition to suggesting questions. In some embodiments, actions can be based on the results of an investigation. For example, if a user conducts an investigation for a particular threat actor and evidence of that threat actor is found, the system can suggest adding a rule so that future actions by the threat actor automatically trigger a detection. In some embodiments, the system can be configured to automatically generate alert configurations for an alerting system. In some embodiments, follow on actions can include, for example, recommending that the user escalate the investigation to a more senior analyst or manager. In some embodiments, the actions that are suggested can include no risk actions, low risk actions, and/or high risk actions. No risk actions can be actions that will have no impact on the production environment, such as executing a query against a data lake. Low risk actions can be actions that will have some impact on the production environment, but which typically do not significantly impact the production environment. For example, in some cases, triggering user reauthentication can be considered a low risk action. In some embodiments, low risk actions can be performed automatically. High risk actions can be actions that will have a significant impact on the production environment. Some examples of high risk actions include closing ports, blocking IP addresses, restricting an asset's network access, locking a user out of their account, and so forth. Typically, high risk actions can be actions that require formal procedures to be followed, such as confirmation by a senior analyst or manager. In some cases, high risk actions can be performed as part of a playbook that details the actions to be taken, the required approvals, and so forth. In some embodiments, the system can be configured to generate a playbook. For example, an LLM can be provided with information

about the action to be taken and optionally any policies or procedures to be followed, and the system can, using the LLM, generate a playbook.

[0121] In the above description, the system is largely described in the context of an investigation that searches for unknown malicious, suspicious, or otherwise unusual activity in an environment. However, the approaches described herein can, additionally or alternatively, be applied in other contexts. For example, in some embodiments, the systems and methods herein can be used to investigate an observed anomaly. For example, in some embodiments, a system can provide a user interface that shows existing alerts. A user can select an alert and can review a summary of information about the alert. In some embodiments, the summary can be generated by an LLM, for example as described herein. The summary can include information such as, for example and without limitation, the specific process, operating system events, and so forth involved in the alert, details about the asset (e.g., machine type, operating system, installed software, version Information, logged in users, remote connections, and so forth). In some embodiments, the summary can include an indication of whether or not an action, such as isolating the asset, was automatically taken. In some embodiments, the summary can provide information about correlated threat intelligence indicators. For example, in some embodiments, the summary can provide information about other assets that were similarly affected, information about ongoing threats, and so forth.

[0122] In some embodiments, the user interface can include a button, link, or other user interface element that a user can select to begin an investigation into the alert using the systems and methods described herein. In some embodiments, an initial question, or a plurality of suggested initial questions, can be based on information about the alert, such as the type of alert, the affected asset, and so forth.

[0123] The systems and methods herein can, additionally or alternatively, be used to guide a user through tasks such as creating new alert rules. In such cases, rather than starting with investigating an alert or conducting a hunt, the user can provide a natural language request to

the system, and the system can, using the LLM and information about query syntax, data lake schema, and/or existing rules, determine an alert syntax that will accomplish the user's goal.

[0124] In some embodiments, when a user chooses to create an alert, for example based on an investigation, the alert can feed back into a related notebook.

[0125] In some embodiments, notebooks can be updated over time. For example, in some embodiments, a user can come back to a notebook after some time and run through it again, ask if there are any updates, and so forth.

[0126] In some embodiments, the systems and methods herein can be used, alternatively or additionally, for issues other than security issues. For example, security monitoring platforms commonly use agents installed on endpoints (e.g., servers, laptops, desktops, tablets, etc.). Misconfiguration of agents can lead to various issues, such as unacceptably high CPU usage, memory usage, storage usage, and so forth. In some cases, misconfigurations can lead to missed alerting, potentially compromising security. In some embodiments, the systems and methods herein can be used to investigate potential configuration issues, such as configuration issues with agents. In some embodiments, a system can be configured to identify issues, for example using telemetry data, much as described herein with respect to identifying and investigating security issues. In some embodiments, the system can identify one or more configuration issues. In some embodiments, an LLM can be used to summarize the issue, provide context for the issue, and/or the like. In some embodiments, the system can suggest steps to take to fix the issue. For example, if an agent is showing high CPU usage, the system can recommend specifying a cap on CPU usage. In some embodiments, the system can suggest one or more next steps. In some embodiments, the system can rank issues and prioritize which issues should be fixed first.

[0127] Figure 3 is a drawing that illustrates an example user interface according to some embodiments. In Figure 3, a user can create a new notebook by clicking the "New Notebook" button. The user interface can show existing notebooks. Existing notebooks can be grouped into personal notebooks ("My Notebooks") and shared notebooks that are shared with and/or by other users. The user interface can include a plurality of buttons that allow a user to select

an investigation type to begin. For example, in Figure 3, a user can select to threat actor, TTP, IOC, asset, malware, or anomaly as a starting point for an investigation. Alternatively, the user can specify a natural language question to begin an investigation. In some embodiments, when a user selects one of the buttons or inputs a natural language question, a system can be configured to create a new notebook for the investigation.

[0128] Figure 4 is a drawing that illustrates another example user interface according to some embodiments. In Figure 4, the user interface shows a log of alerts, for example alerts generating by monitoring software installed on endpoints. The user can select an alert to bring up additional details about the alert. The additional details can be shown in a popup, overlay, etc. The additional details can include alert information, a summary of information related to the alert, and a button to begin an investigation. When a user selects the button to begin the investigation, the system can automatically generate one or more suggested questions based on the alert.

[0129] Figure 5 is a flowchart that illustrates an example process that utilizes retrieval augmented generation (RAG) according to some embodiments. As described herein, RAG can be beneficial for determining relevant contextual information for an input provided by a user. For example, a knowledge base can store information about threat actors, TTPs, IOCs, etc., and the knowledge base can be consulted when generating queries for an LLM. The knowledge base can comprise or can be built from reference documents, which can include blog posts, TTP information, security documentation, and so forth.

[0130] At operation 510, a system can generate LLM embeddings for reference documents 505. The reference documents can include various security-related information such as information about threats, TTPs, IOCs, and so forth. The system can store the generated LLM embeddings for the reference documents 505 in embeddings store 515. At operation 520, the system (or another system) can receive user input, such as a natural language question. Examples of questions can include, “Are there signs of attacks by Lazarus Group on my systems?” or “Are there signs of lateral movements within my network?” Continuing with these examples, the reference documents can include information about specific software,

techniques, etc. used by Lazarus Group and/or information about lateral techniques (e.g., remote services exploitation, remote service session hijacking, RDP hijacking, pass the hash).

[0131] At operation 525, the system can generate an LLM embedding of the received user input. At operation 530, the system can identify relevant reference documents. For example, the system can compare the generated LLM embedding for the received user input to the embeddings stored in the embeddings store 515. The system can identify embeddings in the embeddings store 515 that are most similar to the embedding for the received user input. Various metrics can be used to identify the most similar embeddings, such as cosine similarity, L1 distance, or L2 distances. At operation 535, the system can generate an augmented query based on the received user input and the identified reference documents, which can be fully or partially appended to the user input to generate the augmented query. The augmented query can take various forms, but as an example, if a user asks, "Am I being attacked by Lazarus Group?," an augmented query can be, for example, "Am I being attacked by Lazarus Group? Lazarus Group utilizes access token manipulation, account discovery using Active Directory services, account manipulation by renamed accounts ...." That is, for example, the augmented query can include information about techniques known or believed to be used by Lazarus Group. Thus, even though a user who submits the user input does not necessarily know what techniques are used by Lazarus Group, the system can use RAG to determine what the relevant techniques are for Lazarus Group. At operation 640, the system can provide the augmented query to an LLM.

[0132] In the above description, RAG is provided in the context of reference information about threat actors or techniques. RAG can also be used to aid an LLM in generating queries. For example, to generate queries against a particular database or data lake, the LLM needs to know what language to use for the queries, what the structure/schema of the database or data lake is, and so forth. The language and/or schema can be defined or can be inferred by the LLM by, for example, providing examples of queries (and possible outputs of those queries) to the LLM.

[0133] Figure 6 is a flowchart that illustrates an example process for conducting a threat hunt according to some embodiments. The process shown in Figure 6 can be performed by a computer system. At operation 610, the system can receive a hunt request to begin a new hunt from a user. At operation 615, based on the information supplied by the user (e.g., a type of hunt to conduct), the system can generate one or more suggested questions. In some embodiments, the system can automatically execute a first question. At operation 620, the system can receive a user selection of a question or a user-input natural language question. At operation 625, the system can, by pulling information from a knowledge base, which can include internal and/or external sources, determine one or more relevant IOCs and/or TTPs. At operation 630, the system can provide the IOCs and/or TTPs to an LLM. At operation 635, the system can provide information about a data lake schema and information about a syntax to the LLM. In some embodiments, the LLM may already be trained to generate the appropriate syntax, in which case syntax information may not be provided. In some cases, the LLM may be trained on the schema, in which case schema information may not be provided. At operation 640, the system can use the LLM to generate a query. At operation 645, the system can execute the query against the data lake. At operation 650, the system can provide the results to the user. In some embodiments, providing the results to the user can include providing a summary to the user. In some embodiments, providing the results to the user can include providing contextual information to the user. For example, if the results relate to a particular threat actor, the contextual information can include background information about the threat actor. At operation 655, the system can generate a next suggested question, for example based on previous questions, results of previous questions, etc. The process of generating questions, receiving results, and generating a next question can continue until the investigation is complete, a notebook reaches a maximum length, the number of questions reaches a maximum limit, or a user chooses to end the investigation.

[0134] Figure 7 is a block diagram that illustrates an example process for investigating an alert according to some embodiments. The process illustrated in Figure 7 can be carried out by a computer system. At operation 710, the system can receive an alert selection from a

user. For example, the user can view an alerts dashboard or similar interface and can select an alert. At operation 715, the system can provide summary information to the user. For example, the system can provide information relevant to the alert, such as the type of attack, a potential threat actor, and so forth. At operation 720, the system can receive a user request to begin an investigation. At operation 725, the system can create a new notebook for the investigation. At operation 730, the system can generate one or more suggested questions to begin the investigation. At operation 735, the system can receive a user selection of a question of the one or more suggested questions or can receive a natural language query input by the user.

[0135] At operation 740, the system can determine relevant IOCs and TTPs, for example as described herein. At operation 745, the system can provide the IOCs and TTPs to an LLM. At operation 750, the system can provide schema and syntax information to the LLM. At operation 755, the system can generate a query using the LLM. At operation 760, the system can execute the query against a data lake. At operation 765, the system can provide the results to the user. At operation 770, the system can generate one or more suggested next questions.

[0136] Figure 8 is a block diagram that illustrates an example process for identifying and resolving configuration issues according to some embodiments. The process illustrated in Figure 8 can be performed by a computer system. At operation 810, the system can identify a configuration issue, for example a configuration issue with an agent installed on an endpoint. At operation 820, the system can generate a summary of the issue, for example using an LLM. At operation 830, the system can receive a user request to resolve the configuration issue. At operation 840, the system can determine suggested steps for mitigating the issue. For example, in some embodiments, the system can use an LLM to determine steps for mitigating the issue. In some embodiments, the LLM can use information from a knowledge base, support documents, forums, and so forth in determining suggested steps to resolve the issue.

[0137] In some cases, there may be many configuration issues. Thus, it can be significant to rank the issues so that mitigation can be performed in a prioritized manner. Figure 9 is a block

diagram that illustrates an example process for identifying and ranking issues according to some implementations. The process in Figure 9 can be performed by a computer system. At operation 910, the system can identify a plurality of configuration issues. At operation 920, the system can determine suggested steps to mitigate the issues. At operation 930, the system can determine a ranking for the issues. For example, configuration issues that potentially impact security can have the highest ranking, or configuration issues that impact endpoint performance can have a higher ranking than issues that have limited impact on endpoint performance. At operation 940, the system can provide the ranked issues to a user. At operation 950, the system can provide suggested mitigation steps to the user, for example in response to a user selecting an issue to mitigate.

[0138] In Figure 9, determining issue rankings and determining suggested steps are depicted together. However, it will be appreciated that these can take place separately. For example, a system can be configured to identify and rank issues but may only determine suggested mitigation steps in response to a user request for such steps. This can be significant because, for example, it can reduce the computational loads required to generate steps as users may not choose to fix all issues or may fix issues on multiple endpoints using the same mitigation steps. In some cases, it can take a significant amount of time for users to carry out mitigations. Thus, by generating mitigation steps on demand, the system can help ensure that the provided mitigation steps are not outdated.

[0139] As described herein, after an investigation, it can be beneficial to configure an alert so that future similar observations automatically result in the generation of an alert, thereby reducing the time and effort involved in uncovering or investigating suspicious activity. Accordingly, in some embodiments, at the end of an investigation, the system can be configured to create an alert. In some embodiments, creating an alert may not be done in response to an investigation, but may instead be performed at a user's request without an underlying investigation. For example, a user may provide a natural language prompt such as "Generate an alert to detect Threat Actor XYZ" and the system can, using information in a

knowledge base, generate an LLM prompt, and the LLM can in turn generate appropriate syntax for the alert.

[0140] Figure 10 is a block diagram that illustrates an example process for creating an alert based on an investigation. The process illustrated in Figure 10 can be performed on a computer system. At operation 1010, the computer system can perform an investigation, for example as described herein in cooperation with a user. At operation 1020, the system can determine one or more potential alerts based on the results of the investigation. At operation 1030, the system can receive a user selection to generate an alert. For example, in some embodiments, the system can be configured to provide suggested alerts to the user, and the user can select one or more of the alerts. At operation 1040, the system can generate an LLM prompt to cause an LLM to generate syntax for creating an alert. For example, alerting systems typically have specific syntax that is used to create alerts. In some embodiments, the LLM can be trained and/or provided with information such that the LLM can generate syntactically correct alerting rules. At operation 1050, the system can provide the generated prompt to the LLM. At operation 1060, the system can receive the generated alert syntax. At operation 1070, the system can use the generated alert syntax to create a rule in an alerting system. In some embodiments, prior to creating the rule in the alerting system, the system can prompt the user to confirm that the rule should be created. In some embodiments, the system can provide the generated rule (e.g., a syntactically correct expression of the rule), and the user can then apply the rule using the alerting system.

Computer Systems

[0141] FIG. 11 is a block diagram depicting an embodiment of a computer hardware system configured to run software for implementing automated review of security alerts and any systems or methods disclosed herein. The example computer system 1102 is in communication with one or more computing systems 1120 and/or one or more data sources 1122 via one or more networks 1118. While FIG. 11 illustrates an embodiment of a computing system 1102, it is recognized that the functionality provided for in the components and

modules of computer system 1102 may be combined into fewer components and modules, or further separated into additional components and modules.

[0142] The computer system 1102 can comprise a module 1114 that carries out the functions, methods, acts, and/or processes described herein. The module 1114 is executed on the computer system 1102 by a central processing unit 1106 discussed further below.

[0143] In general, the word "module," as used herein, refers to logic embodied in hardware or firmware or to a collection of software instructions, having entry and exit points. Modules are written in a program language, such as JAVA, C or C++, Python, or the like. Software modules may be compiled or linked into an executable program, installed in a dynamic link library, or may be written in an interpreted language such as BASIC, PERL, Lua, or Python. Software modules may be called from other modules or from themselves, and/or may be invoked in response to detected events or interruptions. Modules implemented in hardware include connected logic units such as gates and flip-flops, and/or may include programmable units, such as programmable gate arrays or processors.

[0144] Generally, the modules described herein refer to logical modules that may be combined with other modules or divided into sub-modules despite their physical organization or storage. The modules are executed by one or more computing systems and may be stored on or within any suitable computer readable medium or implemented in-whole or in-part within special designed hardware or firmware. Not all calculations, analysis, and/or optimization require the use of computer systems, though any of the above-described methods, calculations, processes, or analyses may be facilitated through the use of computers. Further, in some embodiments, process blocks described herein may be altered, rearranged, combined, and/or omitted.

[0145] The computer system 1102 includes one or more processing units (CPU) 1106, which may comprise a microprocessor. The computer system 1102 further includes a physical memory 1110, such as random-access memory (RAM) for temporary storage of information, a read only memory (ROM) for permanent storage of information, and a mass storage device 1104, such as a backing store, hard drive, rotating magnetic disks, solid state disks (SSD),

flash memory, phase-change memory (PCM), 3D XPoint memory, diskette, or optical media storage device. Alternatively, the mass storage device may be implemented in an array of servers. Typically, the components of the computer system 1102 are connected to the computer using a standards-based bus system. The bus system can be implemented using various protocols, such as Peripheral Component Interconnect (PCI), Micro Channel, SCSI, Industrial Standard Architecture (ISA) and Extended ISA (EISA) architectures.

[0146] The computer system 1102 includes one or more input/output (I/O) devices and interfaces 1112, such as a keyboard, mouse, touch pad, and printer. The I/O devices and interfaces 1112 can include one or more display devices, such as a monitor, which allows the visual presentation of data to a user. More particularly, a display device provides for the presentation of GUIs as application software data, and multi-media presentations, for example. The I/O devices and interfaces 1112 can also provide a communications interface to various external devices. The computer system 1102 may comprise one or more multi-media devices 408, such as speakers, video cards, graphics accelerators, and microphones, for example.

[0147] The computer system 1102 may run on a variety of computing devices, such as a server, a Windows server, a Structure Query Language server, a Unix Server, a personal computer, a laptop computer, and so forth. In other embodiments, the computer system 1102 may run on a cluster computer system, a mainframe computer system and/or other computing system suitable for controlling and/or communicating with large databases, performing high volume transaction processing, and generating reports from large databases. The computing system 1102 is generally controlled and coordinated by an operating system software, such as z/OS, Windows, Linux, UNIX, BSD, SunOS, Solaris, MacOS, or other compatible operating systems, including proprietary operating systems. Operating systems control and schedule computer processes for execution, perform memory management, provide file system, networking, and I/O services, and provide a user interface, such as a graphical user interface (GUI), among other things.

[0148] The computer system 1102 illustrated in FIG. 11 is coupled to a network 1118, such as a LAN, WAN, or the Internet via a communication link 1116 (wired, wireless, or a combination thereof). Network 1118 communicates with various computing devices and/or other electronic devices. Network 1118 is communicating with one or more computing systems 1120 and one or more data sources 1122. The module 1114 may access or may be accessed by computing systems 1120 and/or data sources 1122 through a web-enabled user access point. Connections may be a direct physical connection, a virtual connection, and other connection type. The web-enabled user access point may comprise a browser module that uses text, graphics, audio, video, and other media to present data and to allow interaction with data via the network 1118.

[0149] Access to the module 1114 of the computer system 1102 by computing systems 1120 and/or by data sources 1122 may be through a web-enabled user access point such as the computing systems' 1120 or data source's 1122 personal computer, cellular phone, smartphone, laptop, tablet computer, e-reader device, audio player, or another device capable of connecting to the network 1118. Such a device may have a browser module that is implemented as a module that uses text, graphics, audio, video, and other media to present data and to allow interaction with data via the network 1118.

[0150] The output module may be implemented as a combination of an all-points addressable display such as a cathode ray tube (CRT), a liquid crystal display (LCD), a plasma display, or other types and/or combinations of displays. The output module may be implemented to communicate with input devices 1112 and they also include software with the appropriate interfaces which allow a user to access data through the use of stylized screen elements, such as menus, windows, dialogue boxes, tool bars, and controls (for example, radio buttons, check boxes, sliding scales, and so forth). Furthermore, the output module may communicate with a set of input and output devices to receive signals from the user.

[0151] The input device(s) may comprise a keyboard, roller ball, pen and stylus, mouse, trackball, voice recognition system, or pre-designated switches or buttons. The output

device(s) may comprise a speaker, a display screen, a printer, or a voice synthesizer. In addition, a touch screen may act as a hybrid input/output device. In another embodiment, a user may interact with the system more directly such as through a system terminal connected to the score generator without communications over the Internet, a WAN, or LAN, or similar network.

[0152] In some embodiments, the system 1102 may comprise a physical or logical connection established between a remote microprocessor and a mainframe host computer for the express purpose of uploading, downloading, or viewing interactive data and databases on-line in real time. The remote microprocessor may be operated by an entity operating the computer system 1102, including the client server systems or the main server system, an/or may be operated by one or more of the data sources 1122 and/or one or more of the computing systems 1120. In some embodiments, terminal emulation software may be used on the microprocessor for participating in the micro-mainframe link.

[0153] In some embodiments, computing systems 1120 who are internal to an entity operating the computer system 1102 may access the module 1114 internally as an application or process run by the CPU 1106.

[0154] In some embodiments, one or more features of the systems, methods, and devices described herein can utilize a URL and/or cookies, for example for storing and/or transmitting data or user information. A Uniform Resource Locator (URL) can include a web address and/or a reference to a web resource that is stored on a database and/or a server. The URL can specify the location of the resource on a computer and/or a computer network. The URL can include a mechanism to retrieve the network resource. The source of the network resource can receive a URL, identify the location of the web resource, and transmit the web resource back to the requestor. A URL can be converted to an IP address, and a Domain Name System (DNS) can look up the URL and its corresponding IP address. URLs can be references to web pages, file transfers, emails, database accesses, and other applications. The URLs can include a sequence of characters that identify a path, domain name, a file extension, a host name, a query, a fragment, scheme, a protocol identifier, a port number, a

username, a password, a flag, an object, a resource name and/or the like. The systems disclosed herein can generate, receive, transmit, apply, parse, serialize, render, and/or perform an action on a URL.

[0155] A cookie, also referred to as an HTTP cookie, a web cookie, an internet cookie, and a browser cookie, can include data sent from a website and/or stored on a user's computer. This data can be stored by a user's web browser while the user is browsing. The cookies can include useful information for websites to remember prior browsing information, such as a shopping cart on an online store, clicking of buttons, login information, and/or records of web pages or network resources visited in the past. Cookies can also include information that the user enters, such as names, addresses, passwords, credit card information, etc. Cookies can also perform computer functions. For example, authentication cookies can be used by applications (for example, a web browser) to identify whether the user is already logged in (for example, to a web site). The cookie data can be encrypted to provide security for the consumer. Tracking cookies can be used to compile historical browsing histories of individuals. Systems disclosed herein can generate and use cookies to access data of an individual. Systems can also generate and use JSON web tokens to store authenticity information, HTTP authentication as authentication protocols, IP addresses to track session or identity information, URLs, and the like.

[0156] The computing system 1102 may include one or more internal and/or external data sources (for example, data sources 1122). In some embodiments, one or more of the data repositories and the data sources described above may be implemented using a relational database, such as DB2, Sybase, Oracle, CodeBase, and Microsoft® SQL Server as well as other types of databases such as a flat-file database, an entity relationship database, and object-oriented database, and/or a record-based database.

[0157] The computer system 1102 may also access one or more databases 1122. The databases 1122 may be stored in a database or data repository. The computer system 1102 may access the one or more databases 1122 through a network 1118 or may directly access

the database or data repository through I/O devices and interfaces 1112. The data repository storing the one or more databases 1122 may reside within the computer system 1102.

Additional Embodiments

[0158] In the foregoing specification, the systems and processes have been described with reference to specific embodiments thereof. It will, however, be evident that various modifications and changes may be made thereto without departing from the broader spirit and scope of the embodiments disclosed herein. The specification and drawings are, accordingly, to be regarded in an illustrative rather than restrictive sense.

[0159] Indeed, although the systems and processes have been disclosed in the context of certain embodiments and examples, it will be understood by those skilled in the art that the various embodiments of the systems and processes extend beyond the specifically disclosed embodiments to other alternative embodiments and/or uses of the systems and processes and obvious modifications and equivalents thereof. In addition, while several variations of the embodiments of the systems and processes have been shown and described in detail, other modifications, which are within the scope of this disclosure, will be readily apparent to those of skill in the art based upon this disclosure. It is also contemplated that various combinations or sub-combinations of the specific features and aspects of the embodiments may be made and still fall within the scope of the disclosure. It should be understood that various features and aspects of the disclosed embodiments can be combined with, or substituted for, one another in order to form varying modes of the embodiments of the disclosed systems and processes. Any methods disclosed herein need not be performed in the order recited. Thus, it is intended that the scope of the systems and processes herein disclosed should not be limited by the particular embodiments described above.

[0160] It will be appreciated that the systems and methods of the disclosure each have several innovative aspects, no single one of which is solely responsible or required for the desirable attributes disclosed herein. The various features and processes described above may be used independently of one another or may be combined in various ways. All possible combinations and sub-combinations are intended to fall within the scope of this disclosure.

[0161] Certain features that are described in this specification in the context of separate embodiments also may be implemented in combination in a single embodiment. Conversely, various features that are described in the context of a single embodiment also may be implemented in multiple embodiments separately or in any suitable sub-combination. Moreover, although features may be described above as acting in certain combinations and even initially claimed as such, one or more features from a claimed combination may in some cases be excised from the combination, and the claimed combination may be directed to a sub-combination or variation of a sub-combination. No single feature or group of features is necessary or indispensable to each and every embodiment.

[0162] It will also be appreciated that conditional language used herein, such as, among others, "can," "could," "might," "may," "for example," and the like, unless specifically stated otherwise, or otherwise understood within the context as used, is generally intended to convey that certain embodiments include, while other embodiments do not include, certain features, elements and/or steps. Thus, such conditional language is not generally intended to imply that features, elements and/or steps are in any way required for one or more embodiments or that one or more embodiments necessarily include logic for deciding, with or without author input or prompting, whether these features, elements and/or steps are included or are to be performed in any particular embodiment. The terms "comprising," "including," "having," and the like are synonymous and are used inclusively, in an open-ended fashion, and do not exclude additional elements, features, acts, operations, and so forth. In addition, the term "or" is used in its inclusive sense (and not in its exclusive sense) so that when used, for example, to connect a list of elements, the term "or" means one, some, or all of the elements in the list. In addition, the articles "a," "an," and "the" as used in this application and the appended claims are to be construed to mean "one or more" or "at least one" unless specified otherwise. Similarly, while operations may be depicted in the drawings in a particular order, it is to be recognized that such operations need not be performed in the particular order shown or in sequential order, or that all illustrated operations be performed, to achieve desirable results. Further, the drawings may schematically depict one or more

example processes in the form of a flowchart. However, other operations that are not depicted may be incorporated in the example methods and processes that are schematically illustrated. For example, one or more additional operations may be performed before, after, simultaneously, or between any of the illustrated operations. Additionally, the operations may be rearranged or reordered in other embodiments. In certain circumstances, multitasking and parallel processing may be advantageous. Moreover, the separation of various system components in the embodiments described above should not be understood as requiring such separation in all embodiments, and it should be understood that the described program components and systems may generally be integrated together in a single software product or packaged into multiple software products. Additionally, other embodiments are within the scope of the following claims. In some cases, the actions recited in the claims may be performed in a different order and still achieve desirable results.

[0163] Further, while the methods and devices described herein may be susceptible to various modifications and alternative forms, specific examples thereof have been shown in the drawings and are herein described in detail. It should be understood, however, that the embodiments are not to be limited to the particular forms or methods disclosed, but, to the contrary, the embodiments are to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the various implementations described and the appended claims. Further, the disclosure herein of any particular feature, aspect, method, property, characteristic, quality, attribute, element, or the like in connection with an implementation or embodiment can be used in all other implementations or embodiments set forth herein. Any methods disclosed herein need not be performed in the order recited. The methods disclosed herein may include certain actions taken by a practitioner; however, the methods can also include any third-party instruction of those actions, either expressly or by implication. The ranges disclosed herein also encompass any and all overlap, sub-ranges, and combinations thereof. Language such as “up to,” “at least,” “greater than,” “less than,” “between,” and the like includes the number recited. Numbers preceded by a term such as “about” or “approximately” include the recited numbers and should be interpreted based on the

circumstances (for example, as accurate as reasonably possible under the circumstances, for example ±5%, ±10%, ±15%, etc.). For example, “about 10.5 mm” includes “3.5 mm.” Phrases preceded by a term such as “substantially” include the recited phrase and should be interpreted based on the circumstances (for example, as much as reasonably possible under the circumstances). For example, “substantially constant” includes “constant.” Unless stated otherwise, all measurements are at standard conditions including temperature and pressure.

[0164] As used herein, a phrase referring to “at least one of” a list of items refers to any combination of those items, including single members. As an example, “at least one of: A, B, or C” is intended to cover: A, B, C, A and B, A and C, B and C, and A, B, and C. Conjunctive language such as the phrase “at least one of X, Y and Z,” unless specifically stated otherwise, is otherwise understood with the context as used in general to convey that an item, term, etc. may be at least one of X, Y or Z. Thus, such conjunctive language is not generally intended to imply that certain embodiments require at least one of X, at least one of Y, and at least one of Z to each be present. The headings provided herein, if any, are for convenience only and do not necessarily affect the scope or meaning of the devices and methods disclosed herein.

[0165] Accordingly, the claims are not intended to be limited to the embodiments shown herein but are to be accorded the widest scope consistent with this disclosure, the principles and the novel features disclosed herein.

CLAIMS

WHAT IS CLAIMED IS:

1. A method for guided cybersecurity investigation, the method comprising:

generating instructions for presenting a user interface on a user computing device for display to a user;

transmitting the instructions to the user computing device to cause the user computing device to display the user interface;

receiving a request from the user via the user interface to begin a cybersecurity investigation, wherein the cybersecurity investigation is associated with an investigation type;

creating an electronic notebook for the cybersecurity investigation;

determining a question related to the investigation type;

generating a query based on the question type,

wherein the query is generated using a first large language model,

wherein the large language model is configured to augment the question using retrieval augmented generation,

wherein retrieval augment generation is performed against a knowledge base comprising at a schema of a data lake;

storing the query in the notebook;

executing the query against the data lake;

providing results of executing the query to the user by storing at least a portion of the results in the notebook;

analyzing the results of executing the query using a second large language model to generate a summary of the results;

providing the summary of the results to the user by storing the summary in the notebook;

determining, based on the results of executing the query, a recommended next question;

providing the recommended next question to the user; and

receiving a next question from the user, wherein the next question comprises a user-generated question or the recommended next question.

2. The method of claim 1, further comprising:

generating a second query based on the next question;

providing the second query to the user;

executing the second query against the data lake;

providing second results of executing the second query to the user by storing the second results in the notebook;

analyzing the second results of executing the second results using the second large language model to generate a second summary of the second results.

3. A method for guided cybersecurity investigation, the method comprising:

receiving a request from a user via a user interface to begin a cybersecurity investigation, wherein the request comprises a selection of an investigation type;

creating an electronic notebook for the cybersecurity investigation;

determining a first question related to the investigation type;

generating a first query based on the question using a first machine learning model;

executing the first query;

analyzing results of executing the first query using a second machine learning model; and

store the results of executing the first query in the electronic notebook.

4. The method of claim 3, wherein the first machine learning model is configured to utilize retrieval augmented generation to determine the first query.

5. The method of claim 3, wherein the first machine learning model comprises a large language model.

6. The method of claim 3, wherein the investigation type is based on at least one of: threat actor, indicators of compromise (IOCs), tactics, techniques, and procedures (TTPs), entity, malware, campaign, asset, alert, or anomaly.

7. The method of claim 3, wherein the first question is automatically executed.

8. The method of claim 3, wherein generating the first query comprises:

generating an embedding of the first question;

identifying relevant information in a knowledge base based on the embedding of the first question and embeddings of information in the knowledge base, wherein relative information is identified based on a similarity of the embedding of the first question and each embedding of the embeddings of information in the knowledge base; and

generating the first query based on at least the first question and the identified relevant information.

9. The method of claim 3, further comprising:

determining a recommended action; and

providing the recommended action to the user.

10. The method of claim 9, wherein the recommended action comprises creating an alerting rule, wherein the method further comprises:

generating an alert configuration for an alerting system.

11. The method of claim 9, wherein the recommended action comprises a no risk action, a low risk action, or a high risk action.

12. The method of claim 11, wherein the recommended action comprises a low risk action, wherein the low risk action is automatically performed.

13. The method of claim 11, wherein the recommended action comprises a high risk action, wherein the method further comprises:

generating a playbook for performing the high risk action.

14. The method of claim 3, further comprising:

generating a next suggested question, wherein the next suggested question is based on the results of executing the first query.

15. A system for guided cybersecurity investigation, the system comprising:

at least one processor; and

a non-transitory storage medium having instructions thereon that, when executing by the at least one processor, cause the system to:

receive a request from a user via a user interface to begin a cybersecurity investigation, wherein the request comprises a selection of an investigation type;

create an electronic notebook for the cybersecurity investigation;

determine a first question related to the investigation type;

generate a first query based on the question using a first machine learning model;

execute the first query;

analyze results of executing the first query using a second machine learning model; and

store the results of executing the first query in the electronic notebook.

16. The system of claim 15, wherein the first machine learning model is configured to utilize retrieval augmented generation to determine the first query.

17. The system of claim 15, wherein the first machine learning model comprises a large language model.

18. The system of claim 15, wherein the investigation type is based on at least one of: threat actor, indicators of compromise (IOCs), tactics, techniques, and procedures (TTPs), entity, malware, campaign, asset, alert, or anomaly.

19. The system of claim 15, wherein the first question is automatically executed.

20. The system of claim 15, wherein generating the first query comprises:

generating an embedding of the first question;

identifying relevant information in a knowledge base based on the embedding of the first question and embeddings of information in the knowledge base, wherein relative information is identified based on a similarity of the embedding of the first question and each embedding of the embeddings of information in the knowledge base; and

generating the first query based on at least the first question and the identified relevant information.

21. The system of claim 15, further comprising:

determining a recommended action; and

providing the recommended action to the user.

22. The system of claim 21, wherein the recommended action comprises creating an alerting rule, wherein the instructions are further configured to cause the system to:

generate an alert configuration for an alerting system.

23. The system of claim 21, wherein the recommended action comprises a no risk action, a low risk action, or a high risk action.

24. The system of claim 23, wherein the recommended action comprises a low risk action, wherein the low risk action is automatically performed.

25. The system of claim 23, wherein the recommended action comprises a high risk action, wherein the instructions are further configured to cause the system to:

generating a playbook for performing the high risk action.

26. The system of claim 15, wherein the instructions are further configured to cause the system to:

generate a next suggested question, wherein the next suggested question is based on the results of executing the first query.

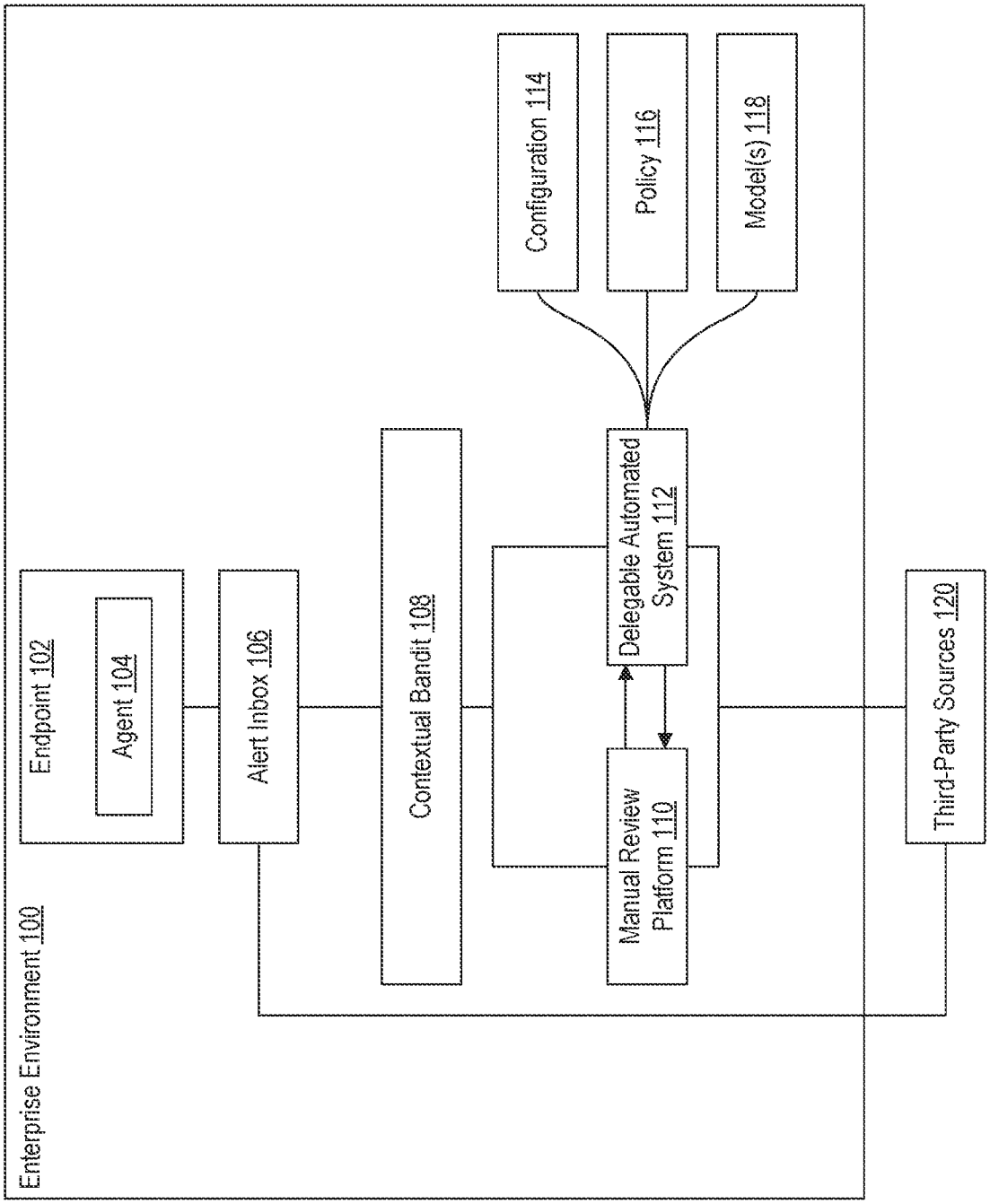
Enterprise Environment 100
Endpoint 102
Agent 104
Alert Inbox 106
Contextual Bandit 108
Manual Review Platform 110
Delegable Automated System 112
Configuration 114
Policy 116
Model(s) 118
Third-Party Sources 120

FIG. 1

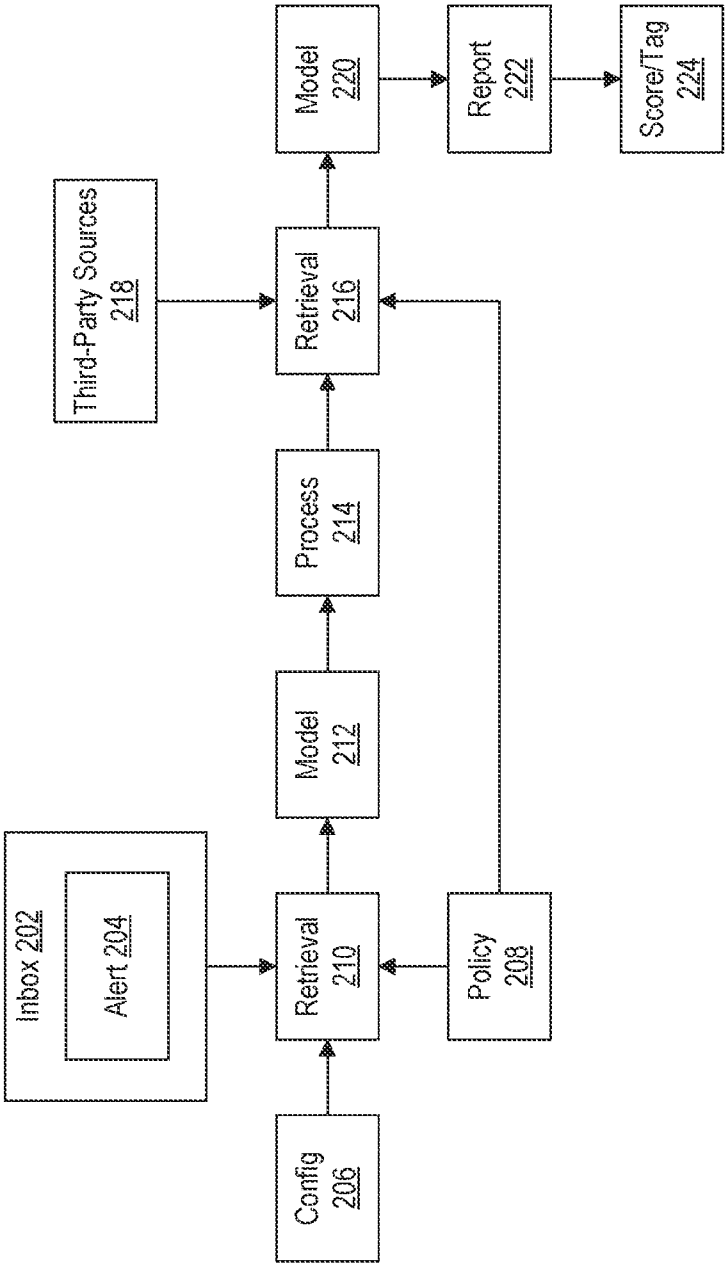
Inbox 202
Alert 204
Config 206
Retrieval 210
Policy 208
Model 212
Process 214
Third-Party Sources 218
Retrieval 216
Model 220
Report 222
Score/Tag 224

FIG. 2

+ New Notebook
Notebooks
My Notebooks
Reverse Engineer critical...
Management threat rep...
Shared Notebooks
APT41 – 2023-04-05 ...
Monthly Report 202 ...
Begin an Investigation
Threat Actor
Profiles cyber attackers and their motives
TTP
Details attackers' methods and strategies
IOC
Identifies signs of security breaches
Asset
Monitors potential cyber attack targets
Malware
Investigates harmful software
Anomaly
Detects unusual patterns signaling threats
Ask a question
SUBMIT

FIG. 3

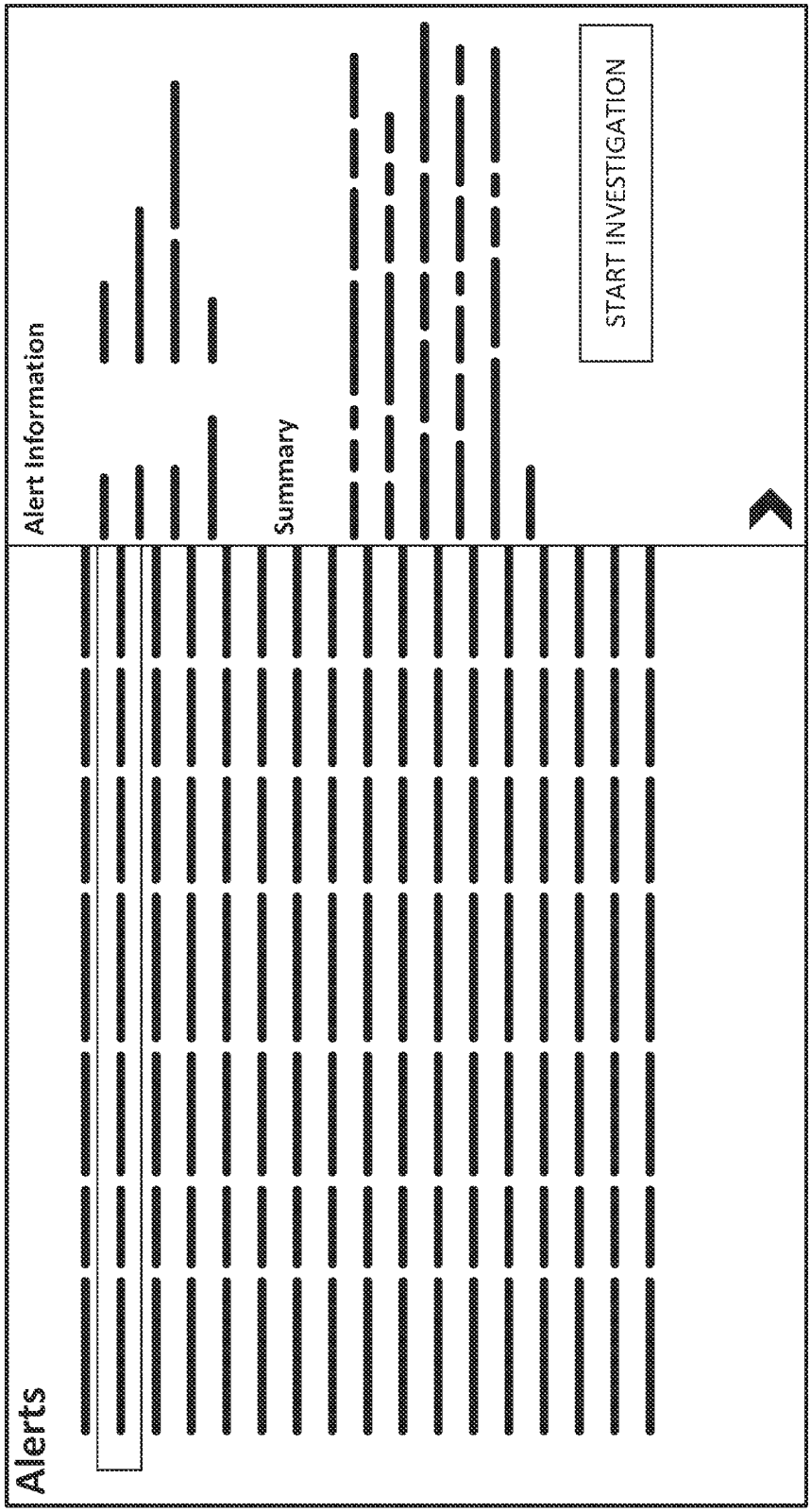

FIG. 4

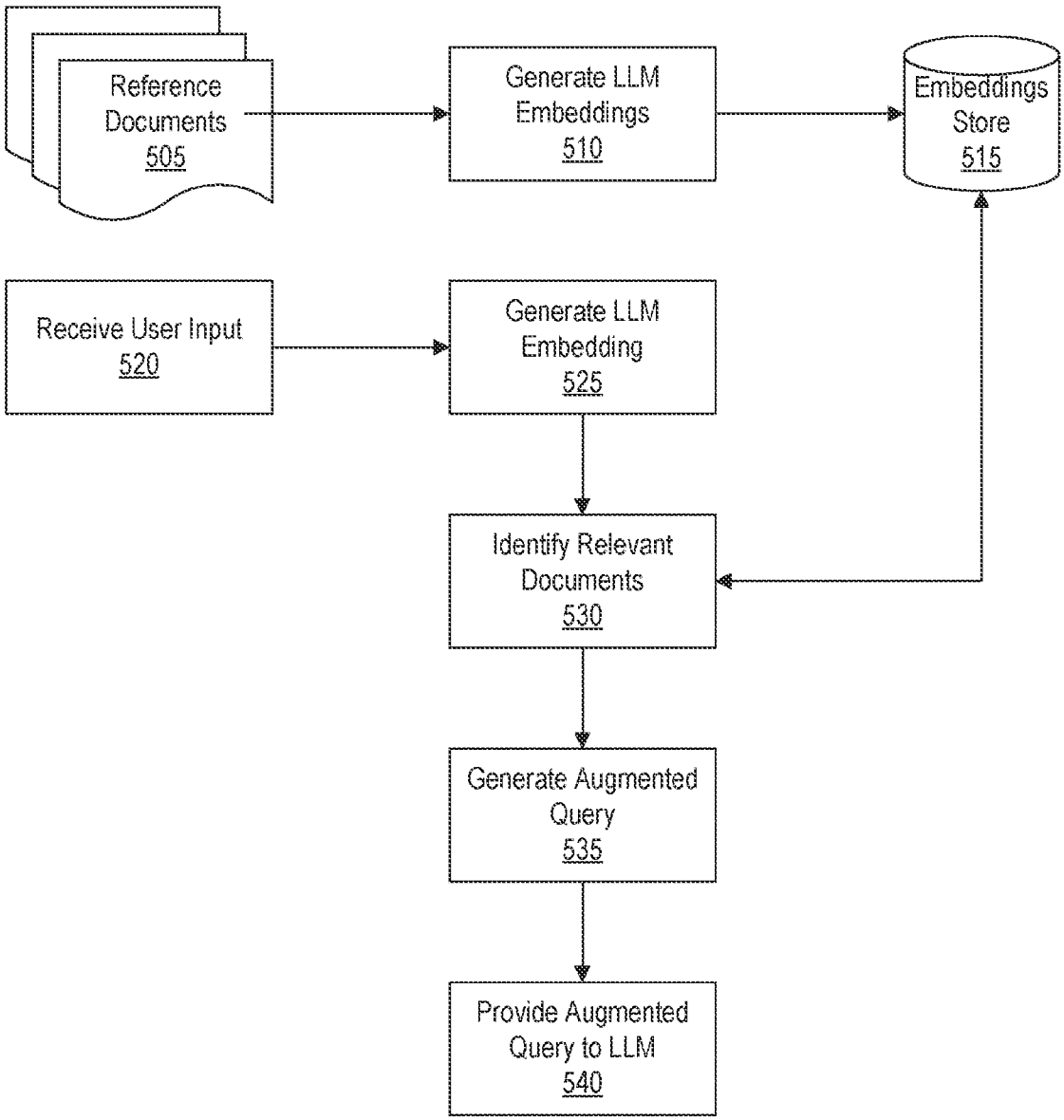
Reference Documents 505
Generate LLM Embeddings 510
Embeddings Store 515
Receive User Input 520
Generate LLM Embedding 525
Identify Relevant Documents 530
Generate Augmented Query 535
Provide Augmented Query to LLM 540

FIG. 5

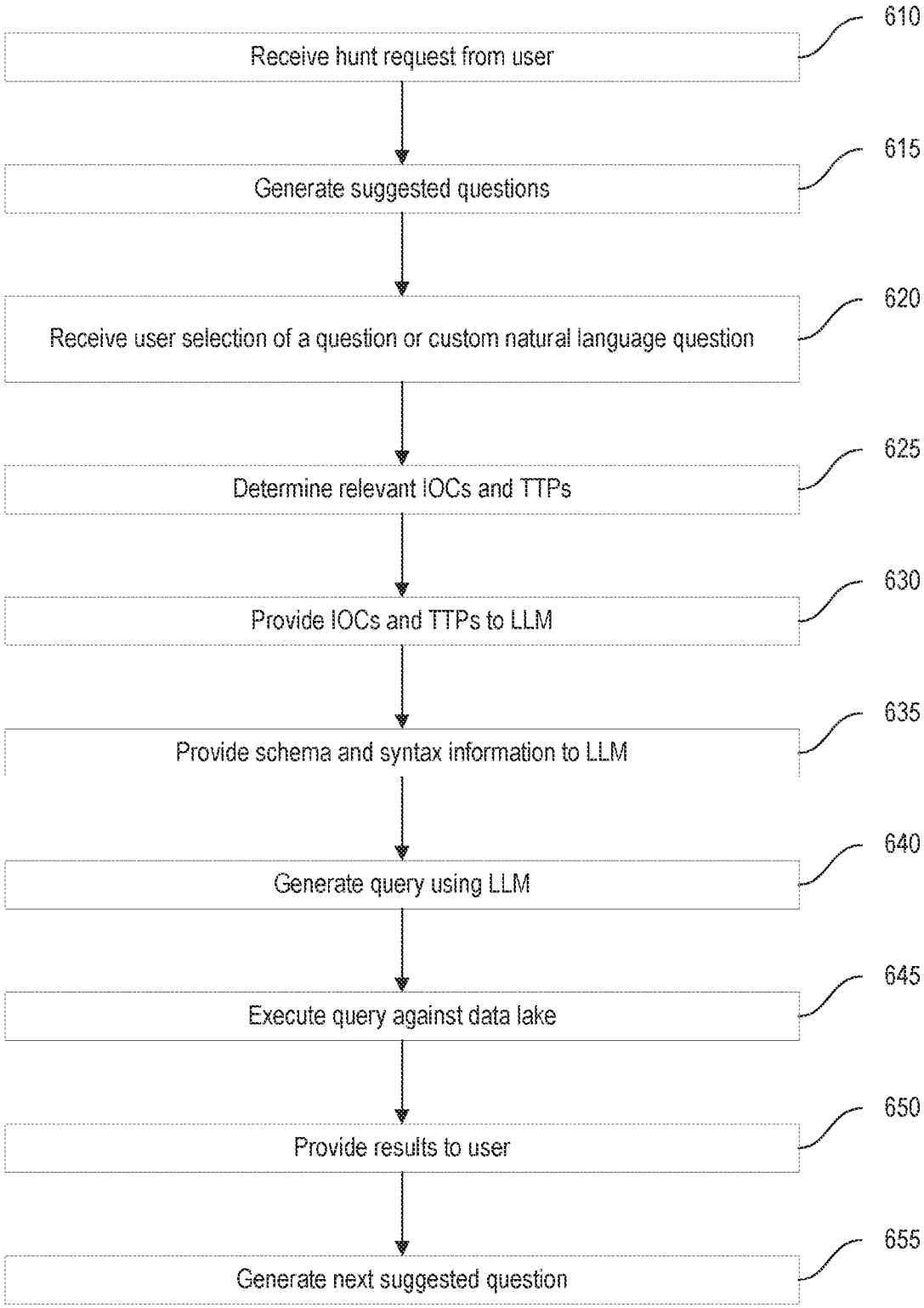
Receive hunt request from user
610
Generate suggested questions
615
Receive user selection of a question or custom natural language question
620
Determine relevant IOCs and TTPs
625
Provide IOCs and TTPs to LLM
630
Provide schema and syntax information to LLM
635
Generate query using LLM
640
Execute query against data lake
645
Provide results to user
650
Generate next suggested question
655

FIG. 6

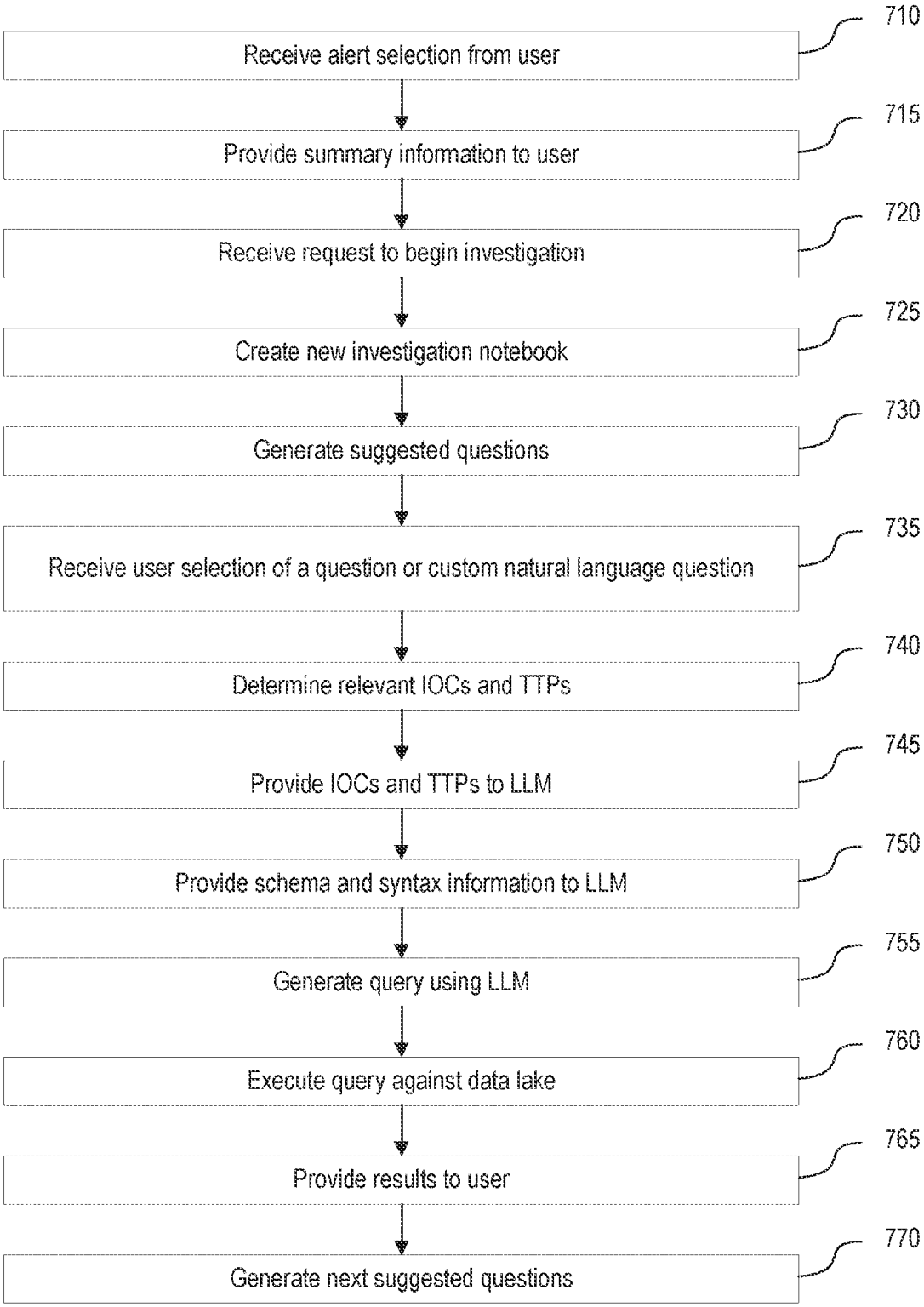
Receive alert selection from user
710
Provide summary information to user
715
Receive request to begin investigation
720
Create new investigation notebook
725
Generate suggested questions
730
Receive user selection of a question or custom natural language question
735
Determine relevant IOCs and TTPs
740
Provide IOCs and TTPs to LLM
745
Provide schema and syntax information to LLM
750
Generate query using LLM
755
Execute query against data lake
760
Provide results to user
765
Generate next suggested questions
770

FIG. 7

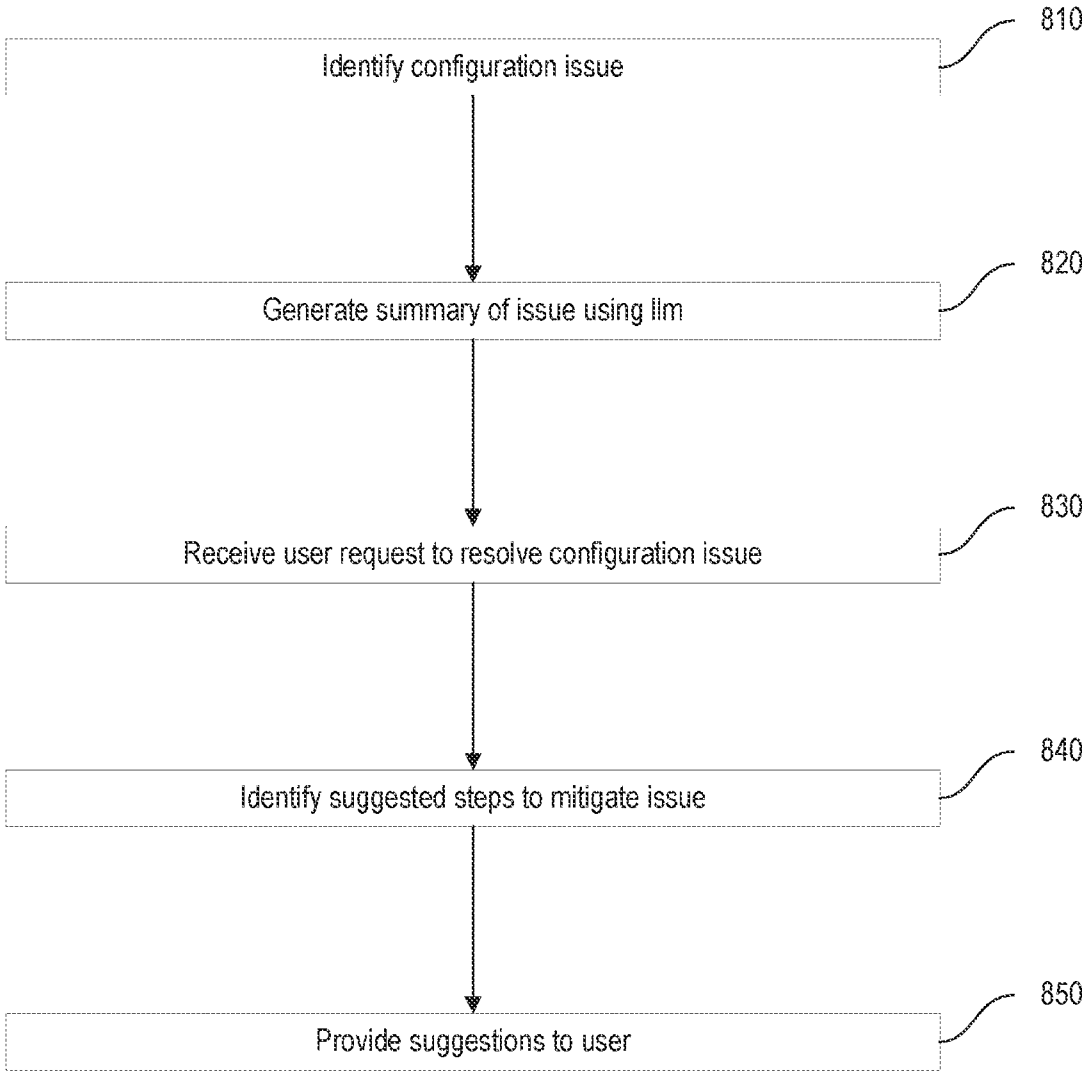
Identify configuration issue
810
Generate summary of issue using llm
820
Receive user request to resolve configuration issue
830
Identify suggested steps to mitigate issue
840
Provide suggestions to user
850

FIG. 8

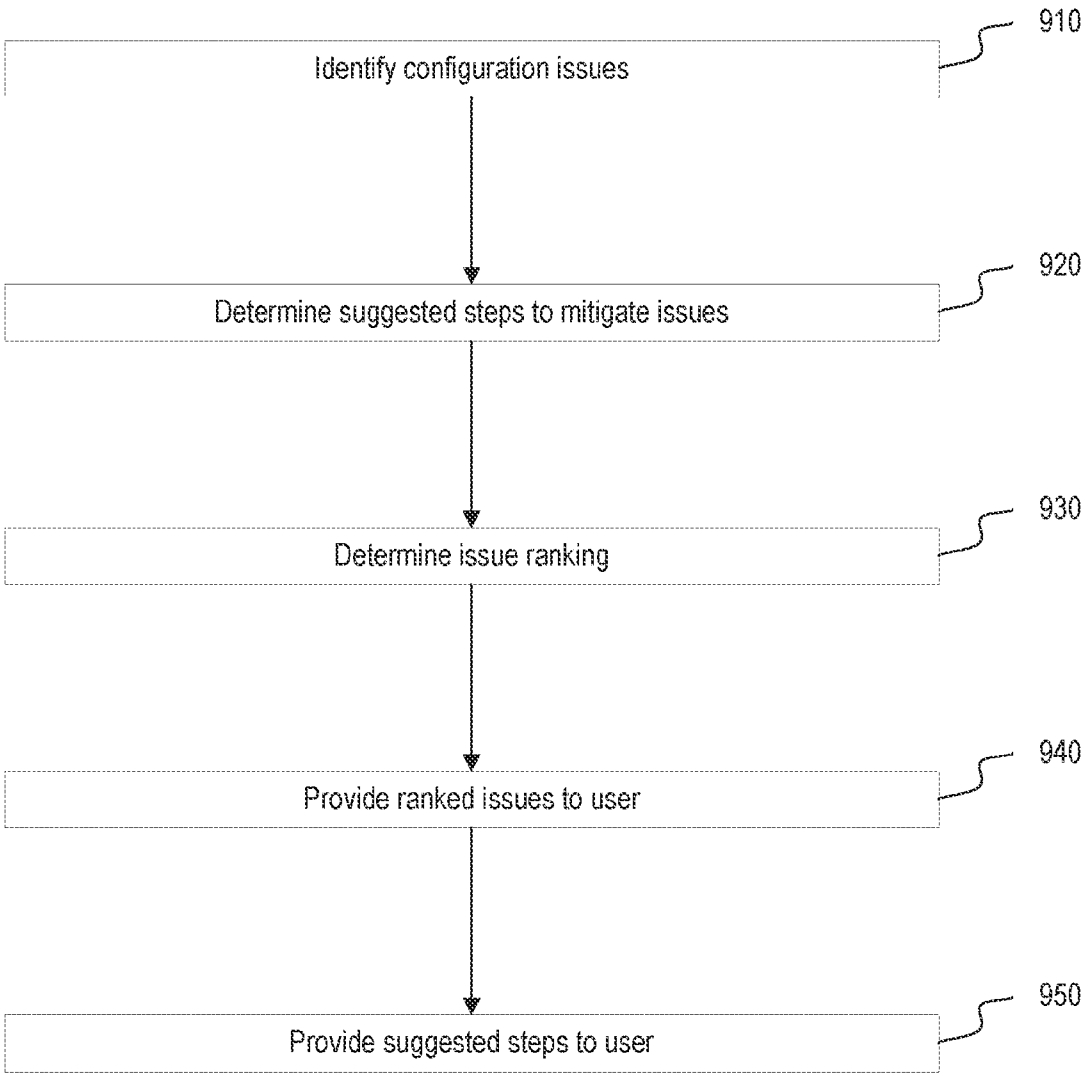
Identify configuration issues
910
Determine suggested steps to mitigate issues
920
Determine issue ranking
930
Provide ranked issues to user
940
Provide suggested steps to user
950

FIG. 9

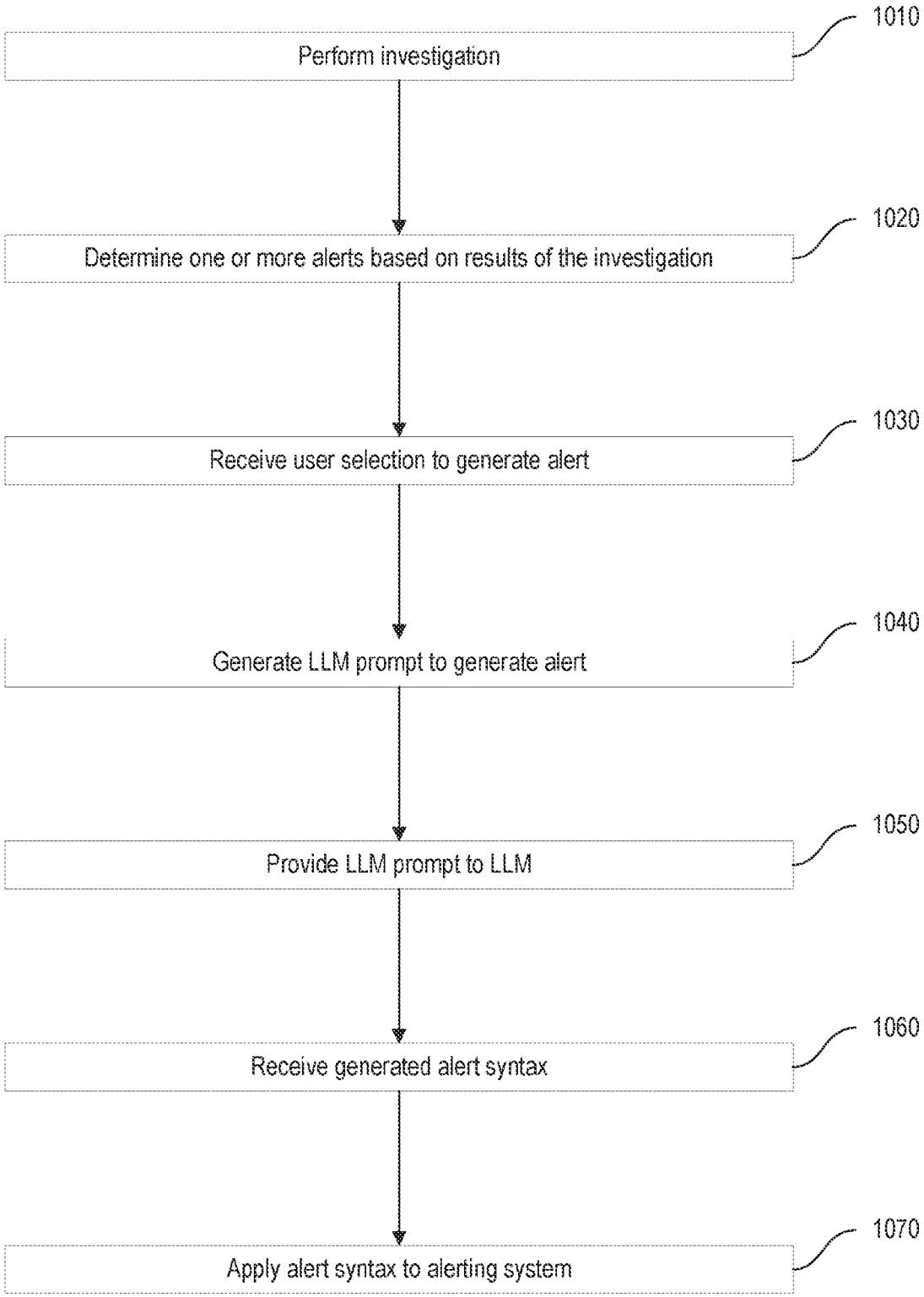
Perform investigation
1010
Determine one or more alerts based on results of the investigation
1020
Receive user selection to generate alert
1030
Generate LLM prompt to generate alert
1040
Provide LLM prompt to LLM
1050
Receive generated alert syntax
1060
Apply alert syntax to alerting system
1070

FIG. 10

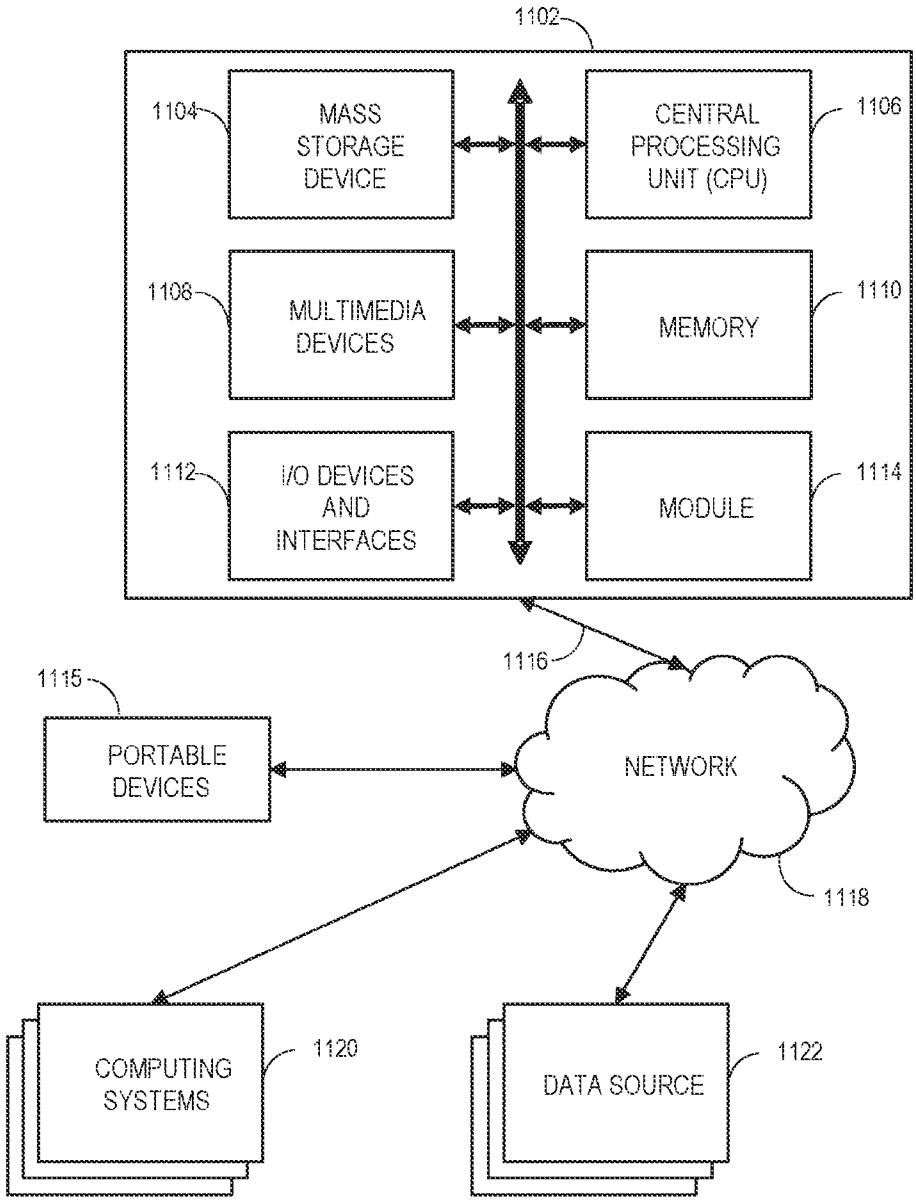
1102
1104
MASS STORAGE DEVICE
CENTRAL PROCESSING UNIT (CPU)
1106
1108
MULTIMEDIA DEVICES
MEMORY
1110
1112
I/O DEVICES AND INTERFACES
MODULE
1114
1116
1115
PORTABLE DEVICES
NETWORK
1118
COMPUTING SYSTEMS
1120
DATA SOURCE
1122

FIG. 11

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2024/051449

A. CLASSIFICATION OF SUBJECT MATTER

IPC: ***G06F 21/55*** (2024.01); ***G06N 3/08*** (2024.01); ***H04L 9/40*** (2024.01); ***G06F 16/2458*** (2024.01)

CPC: ***G06F 21/554; G06F 16/2458; G06N 3/08; H04L 9/40; H04L 63/1416; H04L 63/1425; H04L 63/1433; H04L 63/20***

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

See Search History Document

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

See Search History Document

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

See Search History Document

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2019/0340353 A1 (ENTIT SOFTWARE LLC) 07 November 2019 (07.11.2019) entire document	3-7, 9-19, 21-26
Y	US 2023/0247046 A1 (EXPEL, INC.) 03 August 2023 (03.08.2023) entire document	3-7, 9-19, 21-26
Y	US 2022/0101096 A1 (INTEL CORPORATION) 31 March 2022 (31.03.2022) entire document	4, 16
Y	US 2023/0319089 A1 (CYBEREASON INC.) 05 October 2023 (05.10.2023) entire document	5, 17
Y	US 2020/0274894 A1 (MICROSOFT TECHNOLOGY LICENSING, LLC) 27 August 2020 (27.08.2020) entire document	11-13, 23-25
A	US 2021/0263945 A1 (C3.AI, INC.) 26 August 2021 (26.08.2021) entire document	1-26

☐ Further documents are listed in the continuation of Box C. ☐ See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"D" document cited by the applicant in the international application

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 November 2024 (26.11.2024)	**04 December 2024 (04.12.2024)**

Name and mailing address of the ISA/US	Authorized officer
COMMISSIONER FOR PATENTS **MAIL STOP PCT, ATTN: ISA/US** **P.O. Box 1450** **Alexandria, VA 22313-1450** **UNITED STATES OF AMERICA**	**TAINA MATOS**
Facsimile No. **571-273-8300**	Telephone No. **571-272-4300**