



- (51) **International Patent Classification:**
G06F 17/40 (2006.01) *G06F 21/00* (2006.01)
- (21) **International Application Number:**
PCT/US2015/044903
- (22) **International Filing Date:**
12 August 2015 (12.08.2015)
- (25) **Filing Language:** English
- (26) **Publication Language:** English
- (71) **Applicant:** HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP [US/US]; 11445 Compaq Center Drive West, Houston, TX 77070 (US).
- (72) **Inventors:** KINER, Emil; 1160 Enterprise Way, Sunnyvale, California 94089 (US). WIENER, Guy; Kiryat Technion \$ Technion, 32000 Haifa (IL). HOOLE, Alexander Michael; 1160 Enterprise Way, Sunnyvale, California 94089 (US). SCHMITT, Michael Jason; 1160 Enterprise Way, Sunnyvale, California 94089 (US). O'NEIL, Yekaterina Tsipenyuk; 1160 Enterprise Way, Sunnyvale, California 94089 (US). THOMAS, Dylan B.; 5555 Windward Pkwy, Alpharetta, Georgia 30004 (US). TRAUNER, Daniel Haviv; 1160 Enterprise Way, Sunnyvale, California 94089 (US).
- (74) **Agents:** POSSETT, Ramya et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, CO 80528 (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, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, 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, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

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

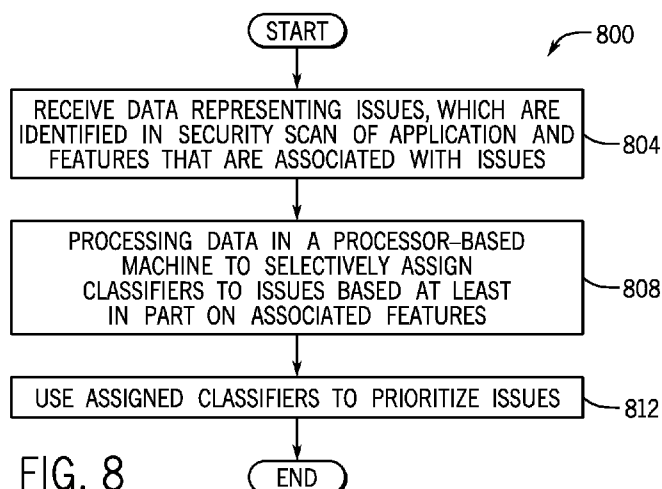
Declarations under Rule 4.17:

- as to the identity of the inventor (Rule 4.17(i))
- as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))

Published:

- with international search report (Art. 21(3))

(54) **Title:** ASSIGNING CLASSIFIERS TO CLASSIFY SECURITY SCAN ISSUES



(57) **Abstract:** A technique includes receiving data representing issues identified in a security scan of an application and features associated with the issues. The technique includes processing the data in a processor-based machine to selectively assign classifiers to the security issues based at least in part on the features. The technique includes using the assigned classifiers to classify the issues.

ASSIGNING CLASSIFIERS TO CLASSIFY SECURITY SCAN ISSUES

BACKGROUND

[001] A given application may have a number of potentially exploitable vulnerabilities, such as vulnerabilities relating to cross-site scripting, command injection or buffer overflow, to name a few. For purposes of identifying at least some of these vulnerabilities, the application may be processed by a security scanning engine, which may perform dynamic and static analyses of the application.

BRIEF DESCRIPTION OF THE DRAWINGS

[002] Fig. 1A is a schematic diagram of a computer system used to prioritize issues identified by an application security scan illustrating the use of human audited issue data to train machine classifiers used by the system according to an example implementation.

[003] Fig. 1B is a schematic diagram of the computer system of Fig. 1A illustrating the use of the machine classifiers of the system to prioritize issues identified in an application security scan according to an example implementation.

[004] Fig. 2 is an illustration of issue data according to an example implementation.

[005] Figs. 3, 5, and 6 are flow diagrams depicting techniques to train machine classifiers according to example implementations.

[006] Fig. 4 is a flow diagram depicting a technique to gather training data according to an example implementation.

[007] Figs. 7 and 8 are flow diagrams depicting techniques to prioritize issues identified by an application security scan using machine classifiers according to example implementations.

[008] Fig. 9 is a schematic diagram of a physical machine according to an example implementation.

DETAILED DESCRIPTION

[009] An application security scanning engine may be used to analyze an application for purposes of identifying potential exploitable vulnerabilities (herein called "issues") of the application. In this manner, the application security scanning engine may provide security scan data (a file, for example), which identifies potential issues with the application, as well as the corresponding sections of the underlying source code (machine-executable instructions, data, parameters being passed in and out of a given function, and so forth), which are responsible for these risks. The application security scanning engine may further assign each issue to a priority bin. In this manner, the application security scanning engine may designate a given issue as belonging to a low, medium, high or critical priority bin, thereby denoting the importance of the issue.

[0010] Each issue that is identified by the application security scanning engine may generally be classified as being either "out-of-scope" or "in-scope." An out-of-scope issue is ignored or suppressed by the end user of the application scan. An in-scope issue is viewed by the end user as being an actual vulnerability that should be addressed.

[0011] There are many reasons why a particular identified issue may be labeled out-of-scope, and many of these reasons may be independent of the quality of the scan output. For example, the vulnerability may not be exploitable/reachable because of environmental mitigations, which are external to the scanned application; the remediation for an issue may be in a source that was not scanned; custom rules may impact the issues returned; and inherent imprecision in the math and heuristics that are used during the analysis may impact the identification of issues.

[0012] In general, the application scanning engine generates the issues according to a set of rules that may be correct, but possibly, the particular security rule that is being applied by the scanning engine may be imprecise. The "out-of-scope" label may be viewed as being a context-sensitive label that is applied by a human auditor. In this manner whether a given issue is out-of-scope, may involve determining whether the issue is reachable and exploitable in this particular application and in this environment, given some sort of external constraints. Therefore, the same issue for two different applications may be considered "in-scope" in one application, but "out-of-scope" in the other; but nevertheless, the identification of the issue may be a "correct" output as

far as the application scanning engine is concerned. In general, human auditing of security scan results may be a relatively highly skilled and time-consuming process, relying on the contextual awareness of the underlying source code.

[0013] One approach to allow the security scanning engine to scan more applications, prioritize results for remediation faster and allow human security experts to spend more time analyzing and triaging relatively high risk issues, is to construct or configure the engine to perform a less thorough scan, i.e., consider a fewer number of potential issues. Although intentionally performing an under-inclusive security scan may result in the reduction of out-of-scope issues, this approach may have a relatively high risk of missing actual, exploitable vulnerabilities of the application.

[0014] In accordance with example implementations that are discussed herein, in lieu of the less thorough scan approach, machine-based classifiers are used to prioritize application security scan results. In this manner, the machine-based classifiers may be used to perform a first order prioritization, which includes prioritizing the issues that are identified by a given application security scan so that the issues are classified as either being in-scope or out-of-scope. The machine-based classifiers may be also used to perform second order prioritizations, such as, for example, prioritizations that involve assigning priorities to in-scope issues. For example, in accordance with example implementations, the machine-based classifiers may assign a priority level of "1" to "6" (in ascending level of importance, for example) to each issue in a given priority bin (priorities may be assigned to issues in the critical priority bin, for example). The machine-based classifiers may also be used to perform other second order prioritizations, such as, for example, reprioritizing the priority bins. For example, the machine-based classifiers may re-designate a given "in-scope" issue as belonging to a medium priority bin, instead of belonging to a critical priority bin, as originally designated by the application security scanning engine.

[0015] In accordance with example implementations that are described herein, the machine-classifiers that prioritize the application security scan results are trained on historical, human audited security scan data, thereby imparting the classifiers with the contextual awareness to prioritize new, unseen application security scan-identified issues for new, unseen applications. More specifically, in accordance with example implementations that are disclosed herein, a given machine classifier is trained to learn the issue preferences of one or multiple human auditors.

[0016] Referring to Fig. 1A, as a more specific example, in accordance with some implementations, a computer system 100 prioritizes application security scan data using machine classifiers 180 (i.e., classification models) and trains the classifiers 180 to learn the issue preferences of human auditors based on historical, human audited application scan data. More specifically, for the example implementation of Fig. 1A, the computer system 100 includes one or multiple on-site systems 110 and an off-site system 160.

[0017] As a more specific example, the off-site system 162 may be a cloud-based computer system, which applies the classifiers 180 to prioritize applicant scan issues for multiple clients, such as the on-site system 110. The clients, such as on-site system 110, may provide training data (derived from human audited application scan data, as described herein) to the off-site system 162 for purposes of training the classifiers 180; and the clients may communicate unaudited (i.e., unlabeled, or unclassified) application security scan data to the off-site system 160 for purposes of using the off-site system's classifiers 180 to prioritize the issues that are identified by the scan data. Depending on the particular implementation, the on-site system 110 may contain a security scanning engine or access scan data is provided by an application scanning engine.

[0018] As depicted in Fig. 1A, the on-site system 110 and off-site system 160 may communicate over network fabric 140, such as fabric associated with wide area network (WAN) connections, local area network (LAN) connections, wireless connections, cellular connections, Internet connections, and so forth, depending on the particular implementation. It is noted that although one on-site system 110 and one off-site system 160 are described herein for an example implementation, the computer system 100 may be entirely disposed at a single geographical location. Moreover, in accordance with further example implementations, the on-site system 110 and/or the off-site system 160 may not be entirely disposed at a single geographical location. Thus, many variations are contemplated, which are within the scope of the appended claims.

[0019] Fig. 1A specifically depicts the communication of data between the on-site system 110 and the off-site system 160 for purposes of training the off-site system's classifiers 180. More specifically, for the depicted example implementation, the on-site system 110 accesses human audited application security scan data 104. In this manner, the human audited application security scan data 104 may be contained in a file that is read by the on-site system 110. The

audited application security scan data 104 contains data that represents one or multiple vulnerabilities, or issues 106, which were identified by an application scanning engine (not shown) by scanning source code of an application.

[0020] In this manner, each issue 106 identifies a potential vulnerability of the application, which may be exploited by hackers, viruses, worms, inside personnel, and so forth. As examples, these vulnerabilities may include vulnerabilities pertaining to cross-site scripting, standard query language (SQL) injection, denial of service, arbitrary code execution, memory corruption, and so forth. As depicted in Fig. 1A, in addition to identifying a particular issue 106, the audited application security scan data 104 may represent a priority bin 107 for each issue 106. For example, the priority bins 107 may be "low," "medium," "high," and "critical" bins, thereby assigning priorities to the issues 106 that are placed therein.

[0021] The audited application security scan data 104 contains data representing the results of a human audit of all or a subset of the issues 106. In particular, the audited application security scan data 104 identifies one or multiple issues 106 as being out-of-scope (via out-of-scope identifiers 108), which were identified by one or multiple human auditors, who performed audits of the security scan data that was generated by the application scanning engine. The audited application security scan data 104 may identify other results of human auditing, such as, for example, reassignment of some of the issues 106 to different priority bins 107 (originally designated by application security scan). Moreover, the audited application security scan data 104 may indicate priority levels for issues 106 in each priority bin 107, as assigned by the human auditors.

[0022] As an example, the audited application security scan data 104 may be generated in the following manner. An application (i.e., source code associated with the application) may first be scanned by an application security scanning engine (not shown) to generate application security scan data (packaged in a file, for example), which may represent the issues 106 and may represent the sorting of the issues 106 into different priority bins 107. Next, one or multiple human auditors may audit the application scan security data to generate the audited application security scan data 104. In this manner, the human auditor(s) may annotate the application security scan data to identify any out-of-scope issues (depicted by out-of-scope identifiers 108 in Fig. 1A), re-designate in-scope issues 106 as belonging to different priority bins 107, assign

priority levels to the in-scope issues 106 in a given priority bin 107, and so forth.

[0023] Each issue 106 has associated attributes, or features, such as one or more of the following (as examples): the identification of the vulnerability, a designation of the priority bin 107, a designation of a priority level within a given priority bin 107, and the indication of whether the issue 106 is in-scope or out-of-scope. Features of the issues 106 such as these, as well as additional features (described herein), may be used to train the classifiers 180 to prioritize the issues 106. More specifically, in accordance with example implementations, as described herein, a classifier 180 is trained to learn a classification preference of a human auditor to a given issue based on features that are associated with the issue.

[0024] Each issue 106 is associated with one or multiple underlying source code sections of the scanned application, called "methods" herein (and which may alternatively be referred to as "functions" or "procedures"). In general, the associated method(s) are the portion(s) of the source code of the application that are responsible for the associated issue 106. A control flow issue is an example of an issue that may be associated with multiple methods of the application.

[0025] In accordance with example implementations, the off-site system 180 trains the classifiers 180 on audited issue data, which is data that represents a decomposition of the audited security scan data 104 into records: each record is associated with one issue 106 and the associated method(s) that are responsible for the issue 106; and each record contains data representing features that are associated with one issue 106 and the associated method(s).

[0026] The issue data may be provided by clients of the off-site system 160, such as the on-site system 110. More specifically, in accordance with example implementations, the on-site system 110 contains a parser engine 112 that processes the audited application security scan data 104 to generate audited issue data 114.

[0027] Referring to Fig. 2 (illustrating the content of the audited issue data 114) in conjunction with Fig. 1A, in accordance with example implementations, the audited issue data 114 contains issue datasets, or records 204, where each record 204 is associated with a given issue 106 and its associated method(s), which are responsible for the issue 106. The record 204 contains data representing features 210 of the associated issue 106 and method(s).

[0028] Depending on the particular implementation, the features 210 may contain 1.) features 212 of the associated issue 106 and method(s), which are derived from the audited

application security scan data 104; and 2.) features 214 of the method(s), which are derived from the source code independently from the application security scan data 104. In this manner, as depicted in Fig. 1A, in accordance with some implementations, the on-site system 110 includes a source code analysis engine 118, which selects source code 120 of the application associated with the method(s) to derive source code metrics 116 (i.e., metrics 116 describing the features of the method(s)), which the parser engine 112 uses to derive the features 214 for the audited issue data 114. In accordance with some implementations, the audited issue data 114 may not contain data representing the features 214.

[0029] As a more specific example, in accordance with some implementations, the features 212 of the audited issue data 114, which are extracted from the audited application security scan data 104, may include one or more of the following: an issue type (i.e., a label identifying the particular vulnerability); a sub-type of the issue 106; a confidence of the application security scanning engine in its analysis; a measure of potential impact of the issue 106; a probability that the issue 106 will be exploited; an accuracy of the underlying rule; an identifier identifying the application security scanning engine; and one or multiple flow metrics (data and control flow counts, data and control flow lengths, and source code complexity, in general, as examples).

[0030] The features 214 derived from the source code 120, in accordance with example implementations, may include one or more of the following: the number of exceptions in the associated method(s); the number of input parameters in the method; the number of statements in the method(s); the presence of a Throw expression in the method(s); a maximal nesting depth in the method(s); the number of execution branches in the method(s), the output type in the method(s), and frequencies (i.e., counts) of various source code constructs.

[0031] In this context, a "source code construct" is a particular programming structure. As examples, a source code construct may be a particular program statement (a Do statement, an Empty Statement, a Return statement, and so forth); a program expression (an assignment expression, a method invocation expression, and so forth); a variable type declaration (a string declaration, an integer declaration, a Boolean declaration and so forth); an annotation; and so forth. In accordance with example implementations, the source code analysis engine 118 may process the source code 120 associated with the method for purposes of generating a histogram

of a predefined set of source code constructs; and the source code analysis engine 118 may provide data to the parser engine 112 representing the histogram. The histogram represents a frequency at which each of its code constructs appears in the method. Depending on the particular implementation, the parser engine 112 may generate audited issue data 114 that includes frequencies of all of the source code constructs that are represented by the histogram or include frequencies of a selected set of source code constructs that are represented by the histogram.

[0032] In accordance with example implementations, the source code analysis engine 118 may generate data that represents control and data flow graphs from the analyzed application and which may form part of the features 214 derived from the source code 120. The properties of these graphs represent the complexity of the source code. As examples, such properties may include the number of different paths, the average and maximal length of these paths, the average and maximal branching factor within these paths, and so forth.

[0033] As described further below, the off-site system 160 uses the audited issue data to train the classifiers 180 so that the classifiers 180 learn the classification preferences of the human auditors for purposes of prioritizing the issues 106. More specifically, referring to Fig. 3, in accordance with example implementations, a technique 300 to train a given security scan classifier includes receiving (block 304) data representing an output of a security scan of an application and an audit of the output of the security scan by a human auditor. The output represents an issue with the application, which is identified by the security scan, and the audit represents an analysis of the issue by the human auditor. The technique 300 includes training (block 308) the classifier to learn an issue preference of the human auditor. The training includes processing data in a processor-based machine to, based at least in part on the output of the security scan and the analysis of the security scan by the human auditor, learn the classification preference of the human auditor to the issue to build a classification model for the issue.

[0034] Referring back to Fig. 1A, in accordance with example implementations, the classifiers 180 may be trained using anonymized data. In this manner, in accordance with example implementations, data communicated between the on-site system 110 and off-site system 160 is anonymized, or sanitized, to remove labels, data and so forth, which may reveal confidential or business sensitive information, the associated entity providing the application,

users of the application, and so forth. Due to the anonymization of human audited data scan, the off-site system 160 may gather a relatively large amount of training data for its classifiers 180 from clients that are associated with different business entities and different application products. Moreover, this approach allows collection of training data that is associated with a relatively large number of programming languages, source code constructs, human auditors, and so forth, which may be beneficial for training the classifiers 180, as further described herein.

[0035] As depicted in Fig. 1A, in accordance with example implementations, an anonymization engine 130 may sanitize the audited issue data 114 to provide anonymized audited issue data 132, which may be communicated via the network fabric 140 to the off-site system 160. In accordance with example implementations, the off-site system 160 may include a job manager engine 162, which among its responsibilities, controls routing of the anonymized audited issue data 132 to a data store 166. In this regard, in accordance with example implementations, the off-site system 160 collects anonymized audited issue data (such as data 132) from multiple, remote clients (such as on-site system 110) for purposes of training the classifiers 180. In accordance with further example implementations, the parser engine 112 may provide anonymized data, and the on-site system 110 may not include the anonymization engine 130.

[0036] Referring to Fig. 4, thus, in accordance with example implementations, a technique 400 to gather training data for the classifiers 180 includes receiving (block 404) audited application security scan data and associated source code. For each issue and method combination, the technique includes parsing audited security scan data to extract data corresponding to features that are associated with the issue and method and determining features for the method based on the source code to derive issue data, pursuant to block 408. The issue data contains a record of the features, for each issue and method combination. Pursuant to block 416, the method 300 includes anonymizing the issue data, and the anonymized data is stored (block 420) for future classifier training.

[0037] In accordance with example implementations, each classifier 180 is associated with a training policy. Each training policy, in turn, may be associated with a set of filtering parameters 189, which define filtering criteria for selecting training data that corresponds to specific issue attributes, or features, which are to be used to train the classifier 180. In

accordance with example implementations, to train a given classifier 180, a training engine 170 of the off-site system 160 selects the set of filter parameters 189 based on the association of the set to the training policy of the classifier 180 to select specific, anonymized audited issue data 172 (Fig. 1A) to be used in the training. Using the selected anonymized issue data 172, the training engine 170 applies a machine learning algorithm to build a classification model for the classifier 180. Depending on the particular implementation, the training engine 170 may be different training policies for all classifiers 180 or may use different training policies for different groups of classifiers 180. Depending on the particular implementation, the training engine 170 may build one of the following classification models (as examples) for the classifiers 180: a support vector machine (SVM) model, a neural network model, a decision tree model, ensemble models, and so forth.

[0038] The selected anonymized audited issue data 172 thus, focuses on specific records 204 of the anonymized issue data 132 for training a given classifier 180, so that the classifier 180 is trained on the specific classification preference(s) of the human auditor(s) for the corresponding issue(s) to build a classification model for the issue(s). Thus, referring to Fig. 5, in accordance with example implementations, a technique 500 to train a classifier includes selecting (block 504) a training policy for the classifier and determining (block 508) one or multiple filtering parameters based on the selected training policy. The method 500 includes filtering (block 512) stored issue data with the selected filtering parameter(s) to select one or multiple records for training the classifier and training (block 516) the classifier on features represented by data contained in the selected record(s).

[0039] Other ways may be used to select record(s) for training a given classifier 180, in accordance with further implementations. For example, in accordance with another example implementation, an attribute-to-training policy mapping may be applied to the records 204 to map the issue records to corresponding training policies (and thus, map the records 204 to the classifiers 180 that are trained with the records 204). Thus, in general, a technique 600 (see Fig. 6) to train a classifier, in accordance with example implementations, includes accessing (block 604) issue datasets. Each issue dataset represents an issue with a portion of source code, which is identified by the security scanning of the source code, attributes (or features) associated with the scanning, and a result of human auditing of the issue. The technique 600 includes selecting

(block 608) a given dataset to train the classifier based at least in part on the attributes and using (block 612) the selected dataset (among other datasets selected in the same manner, for example) to train the classifier.

[0040] Fig. 1B illustrates data flows of the computer system 100 for purposes of classifying unaudited application security scan data 190 (i.e., the output of an application security scanning engine) to produce corresponding machine classified application security scan data 195. In this manner, the unaudited application security scan data 190 and the classified application security scan data 195 both identify issues 106, which were initially identified by an application security scan. The classified application security scan data 195 contains data representing a machine-classified-based prioritization of the security scan. In this manner, the classified application security scan data 195 may identify out-of-scope issues (as depicted by out-of-scope identifiers 197), priority bins 107 for the in-scope issues 106, priorities for the in-scope issues 106 of a given priority bin 107, and so forth.

[0041] More specifically, for the classification to occur, in accordance with some implementations, the parser engine 112 parses the unaudited application security scan data 190 to construct unclassified issue data 115. In accordance with example implementations, similar to the audited issue data 114 discussed above in connection with Fig. 1A, the unclassified issue data 110 is arranged in records; each record is associated with a method and issue combination; and each record contains data representing features derived from the application security scan data 190. Moreover, depending on the particular implementation, each record may also contain data representing features derived from the associated source code 120.

[0042] As depicted in Fig. 1B, the anonymization engine 130 of the on-site system 110 sanitizes the unclassified issue data 115 to provide anonymized unclassified issue data 133. The anonymized unclassified issue data 133, in turn, is communicated from the on-site system 110 to the off-site system 160 via the network fabric 140. As depicted in Fig. 1B, the job manager engine 162 routes the anonymized unclassified issue data 133 to the classification engine 182.

[0043] In accordance with example implementations, each classifier 180 is associated with a classification policy, which defines the features, or attributes, of the issues that are to be classified by the classifier 180. Moreover, in accordance with example implementations, the classification engine 182 may apply an attribute-to-classifier mapping 191 to the anonymized

classified issue data 183 for purposes of sorting the records 204 of the data 182 according to the appropriate classification policies (and correspondingly sort the records 204 to identify the appropriate classifiers 180 to be applied to prioritize the results).

[0044] The classification engine 182 applies the classifiers 180 to the records 204 that conform to the corresponding classification policies. Thus, by applying the attribute-to-classification policy mapping 191 to the anonymized unclassified issue data 133, the classification engine 182 may associate the records of the data 133 with the predefined classification policies and apply the corresponding selected classifiers 182 to the appropriate records 204 to classify the records. This classification results in anonymized classified issue data 183. The anonymized classified issue data 183, in turn, may be communicated via the network fabric 140 to the on-site system 110 where the data 183 is received by the parser engine 112. In accordance with example implementations, the parser engine 112 performs a reverse transformation anonymized of the classified issue data 183, de-anonymizes the data and arranges the data in the format associated with the output of the security scanning engine to provide the classified application security scan data 195.

[0045] Thus, in accordance with example implementations, a technique 700 (see Fig. 7) to classify an issue associated with a given record 204 includes receiving (block 704) and application security scan data and associated source code. The technique 700 includes parsing (block 506) the application security scan data to provide issue data, which contains a record for each issue and method combination; and the technique 700 includes applying (block 708) an attribute-to-classification policy mapping to the issue data, resulting in identification of a classification category for the given record. A classifier may then be selected (block 712) based on the identified classification policy, and the selected classifier may be used (block 716) to prioritize the issue that is associated with the given record.

[0046] Other ways may be used to select a classifier 180 for prioritizing a given issue, in accordance with further implementations. For example, in accordance with another example implementation, the issue data may be filtered through different filters (each being associated with a different classification policy) for purposes of associating the records with classification policies (and classifiers 180). Thus, in general, a technique 800 (see Fig. 8) to classify an issue includes receiving (block 804) data representing an issues, which was identified in a security

scan in an application and features that are associated with the issue; and processing (block 808) the data in a processor-based machine to selectively assign a classifier to the issues based at least in part on the features. The technique 800 includes using the assigned classifiers to classify the issue, pursuant to block 812.

[0047] A given training policy or classification policy may be associated with one or multiple issue features. For example, a given classification policy may specify that an associated classifier 180 is to be used to prioritize issues that have a certain set of features; and likewise a given training policy for a classifier 180 may specify that an associated classifier is to be trained on issue data having a certain set of features. It is noted that, in accordance with example implementations, it is not guaranteed that the issue attribute-to-classifier mapping corresponds to the sum total of the training policies of the relevant classifiers 180. This allows for the classification policy for a given classifier 180 to allow an issue record to be used for a given the classifier 180 for classification purposes, even though that issue's attributes (and thus, the record) may be excluded for training of the classifier 180 by the classifier's training policy.

[0048] As a more specific example, a particular classification or training policy may be associated with an issue type and the identification (ID) of a particular human auditor who may be preferred for his/her classification of the associated issue type. In this manner, the skills of a particular human auditor may highly regarded for purposes of classifying a particular issue/method combination due to the auditor's overall experience, skill pertaining to the issue or experience with a particular programming language.

[0049] The classification or training policy may be associated with characteristics other than a particular human auditor ID. For example, the classification or training policy may be associated with one or multiple characteristics of the method(s). The classification or training policy may be associated with one or multiple features pertaining to the degree of complexity of the method. The classification or training policy may be associated with methods that exceed or are below a particular data or control flow count threshold; exceed or are below a particular data or control length threshold; exceed or are below a count threshold for a collection of selected source code constructs; have a number of exceptions that exceed or are below a threshold; have a number of branches that exceed or are below a threshold; and so forth. As another example, the classification or training policy may be associated with the programming language associated

with the method(s).

[0050] As other examples, the classification or training policy may be associated with one or multiple characteristics of the application security scanning engine. For example, the classification or training policy may be associated with a particular ID, date range, or version of the application security engine. The classification or training policy may be associated with one or multiple characteristics of the scan, such as a particular date range when the scan was performed; a confidence assessed by the application scanning engine within a particular range of confidences, an accuracy of the scan within a particular range of accuracies; a particular ID, date range, or version of the application security engine; and so forth. Moreover, the classification or training policy may be associated with an arbitrary feature, which is included in the record and is specified by a customer.

[0051] As a more specific example, a particular classification or training policy may be associated with the following characteristics that are identified from the features or attributes of the issue record: Human Auditor A, the Java programming language, an application security scan that was performed in the last two years, and a specific issue type (a flow control issue, for example).

[0052] Referring to Fig. 9 in conjunction with Fig. 1A, in accordance with example implementations, the on-site system 110 and/or off-site system 160 may each have an architecture that is similar to the architecture that is depicted in Fig. 9. In this manner, the architecture may be in the form of a system 900 that includes one or more physical machines 910 (N physical machines 910-1...9-10-N, being depicted as examples in Fig. 9). The physical machine 910 is an actual machine that is made up of actual hardware 920 and actual machine executable instructions 950. Although the physical machines 910 are depicted in Fig. 9 as being contained within corresponding boxes, a particular physical machine may be a distributed machine, which has multiple nodes that provide a distributed and parallel processing system.

[0053] In accordance with exemplary implementations, the physical machine 910 may be located within one cabinet (or rack); or alternatively, the physical machine 910 may be located in multiple cabinets (or racks).

[0054] A given physical machine 910 may include such hardware 920 as one or more processors 914 and a memory 921 that stores machine executable instructions 950, application

data, configuration data and so forth. In general, the processor(s) 914 may be a processing core, a central processing unit (CPU), and so forth. Moreover, in general, the memory 921 is a non-transitory memory, which may include semiconductor storage devices, magnetic storage devices, optical storage devices, and so forth. In accordance with example implementations, the memory 921 may store data representing the data store 166 and data representing the one or more classifiers 180 (i.e., classification models). The data store and/or classifiers 180 may be stored in another type of storage device (magnetic storage, optical storage, and so forth), in accordance with further implementations.

[0055] The physical machine 910 may include various other hardware components, such as a network interface 916 and one or more of the following: mass storage drives; a display, input devices, such as a mouse and a keyboard; removable media devices; and so forth.

[0056] For the example implementation in which the system 900 is used for the off-site system 160 (depicted in Fig. 9), the machine executable instructions 950 may, when executed by the processor(s) 914, cause the processor(s) 914 to form one or more of the job manager engine 162, training engine 170 and classification engine 182. It is noted that although Fig. 9 depicts an example implementation for the off-site system 160, for example implementations in which the system 900 is used for the on-site system 110, the machine-executable instructions 950 may, when executed by the processor(s) 914, cause the processor(s) 914 to form one or more of the parser engine 112, source code analysis engine 118 and anonymization engine 130.

[0057] In accordance with further example implementations, one or more of the components of the off-site system 160 and/or on-site system 110 may be constructed as a hardware component that is formed from dedicated hardware (one or more integrated circuits, for example). Thus, the components may take on one or many different forms and may be based on software and/or hardware, depending on the particular implementation.

[0058] In general, the physical machines 910 may communicate with each other over a communication link 970. This communication link 970, in turn, may be coupled to the network fabric 140 and may contain one or more multiple buses or fast interconnects.

[0059] As an example, the system 900 may be an application server farm, a cloud server farm, a storage server farm (or storage area network), a web server farm, a switch, a router farm, and so forth. Although two physical machines 910 (physical machines 910-1 and 910-N) are

depicted in Fig. 9 for purposes of a non-limiting example, it is understood that the system 900 may contain a single physical machine 910 or may contain more than two physical machines 910, depending on the particular implementation (i.e., "N" may be "1," "2," or a number greater than "2").

[0060] While the present techniques have been described with respect to a number of embodiments, it will be appreciated that numerous modifications and variations may be applicable therefrom. It is intended that the appended claims cover all such modifications and variations as fall within the scope of the present techniques.

WHAT IS CLAIMED IS:

- 1 1. A method comprising:
2 receiving data representing issues identified in a security scan of an application and
3 features associated with the issues;
4 processing the data in a processor-based machine to selectively assign classifiers to the
5 security issues based at least in part on the features; and
6 using the assigned classifiers to classify the issues.
- 1 2. The method of claim 1, wherein each of the issues is associated with a group of the
2 features, and processing the data to selectively assign the classifiers comprises filtering the
3 groups of features based on at least one filtering parameter associated with classification
4 category to identify at least one issue to be assigned to a classifier associated with the
5 classification category.
- 1 3. The method of claim 2, wherein the filtering comprises filtering to identify an issue to be
2 classified by a classifier trained on data associated with a specific human auditor.
- 1 4. The method of claim 2, wherein the filtering comprises filtering to identify an issue to be
2 classified by a classifier trained on data associated with a similar application type.
- 1 5. The method of claim 1, wherein the features comprise features identified in the
2 application security scan.
- 1 6. The method of claim 1, wherein the features comprise metrics derived from source code
2 of the application associated with the issues.

- 1 7. An apparatus comprising:
2 a data store;
3 a first engine comprising a processor to receive human audited issue datasets associated
4 with a plurality of application security scans and store the datasets in the data store, wherein each
5 issue dataset represents issues identified in the associated application security scan and each
6 issue of the associated dataset has associated attributes; and
7 a second engine comprising a processor to train a classifier, wherein the second engine:
8 selects a subset of the issue datasets based at least in part on an attribute-to-
9 classification category mapping;
10 retrieves the selected subset of issue datasets from the data store; and
11 uses the selected subset of issue datasets to train the classifier.
- 1 8. The apparatus of claim 7, wherein the attribute-to-classification category mapping groups
2 the issue datasets according to categories of applications.
- 1 9. The apparatus of claim 7, wherein each audited issue dataset is associated with a human
2 auditor of a plurality of human auditors associated with the issue datasets, and the second engine
3 uses the attribute-to-classification category mapping to select at least one issue dataset associated
4 with a given human auditor.
- 1 10. The apparatus of claim 7, wherein each audited issue dataset is associated with a
2 programming language of a plurality of programming languages associated with the issue
3 datasets, and second engine uses the attribute-to-classification category mapping to select at least
4 one issue dataset associated with a given programming language.
- 1 11. The apparatus of claim 7, wherein at least some of the audited issue datasets are each
2 associated with a project for an entity of a plurality of projects for the entity, and second engine
3 uses the attribute-to-classification category mapping to select at least one issue dataset associated
4 with a given project for the entity.

1 12. An article comprising a non-transitory computer readable storage medium to store
2 instructions that when executed by a processor-based machine cause the processor-based
3 machine to:

4 receive data representing a human audited output of a security scan of an application, the
5 output identifying a plurality of security issues with the application; and

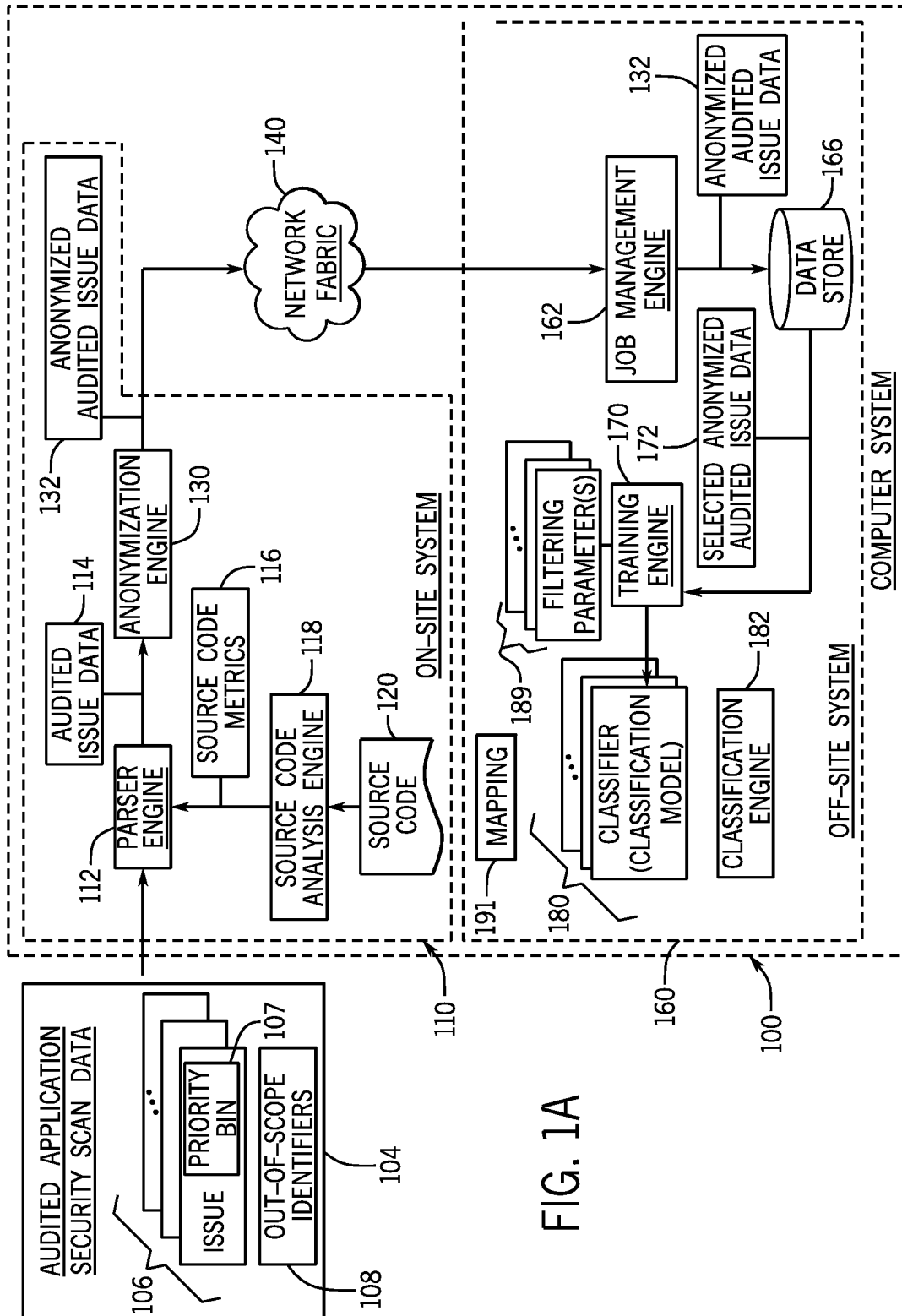
6 select a classifier to classify a given issue of the plurality of security issues from a
7 plurality of candidate classifiers based at least in part on at least one attribute of the given issue.

1 13. The article of claim 12, wherein the given issue is associated with a set of attributes
2 including the at least one attribute, and the storage medium storing instructions that when
3 executed by the processor-based system cause the processor-based system to apply at least one
4 filter to the set of attributes to identify the selected classifier from the plurality of candidate
5 classifiers.

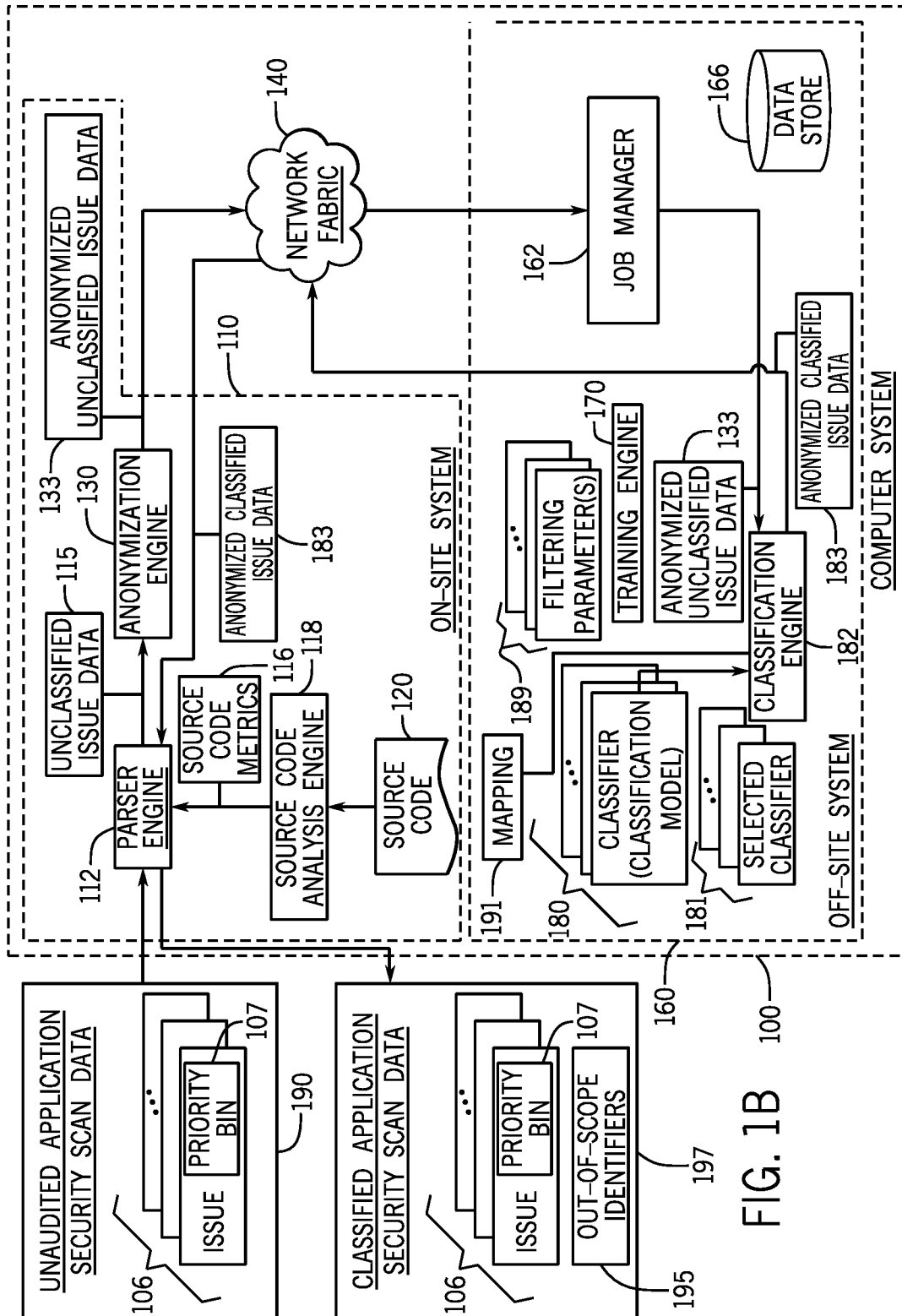
1 14. The article of claim 13, wherein the at least filters based at least on one or more of the
2 following: a programming language, a human auditor identity, an issue severity, an entity
3 associated with the application, a date range and an attribute defined by the entity associated with
4 the application.

1 15. The article of claim 12, wherein the plurality of candidate classifiers are trained based on
2 anonymized security scan outputs associated with a plurality of entities.

1 / 9



2 / 9



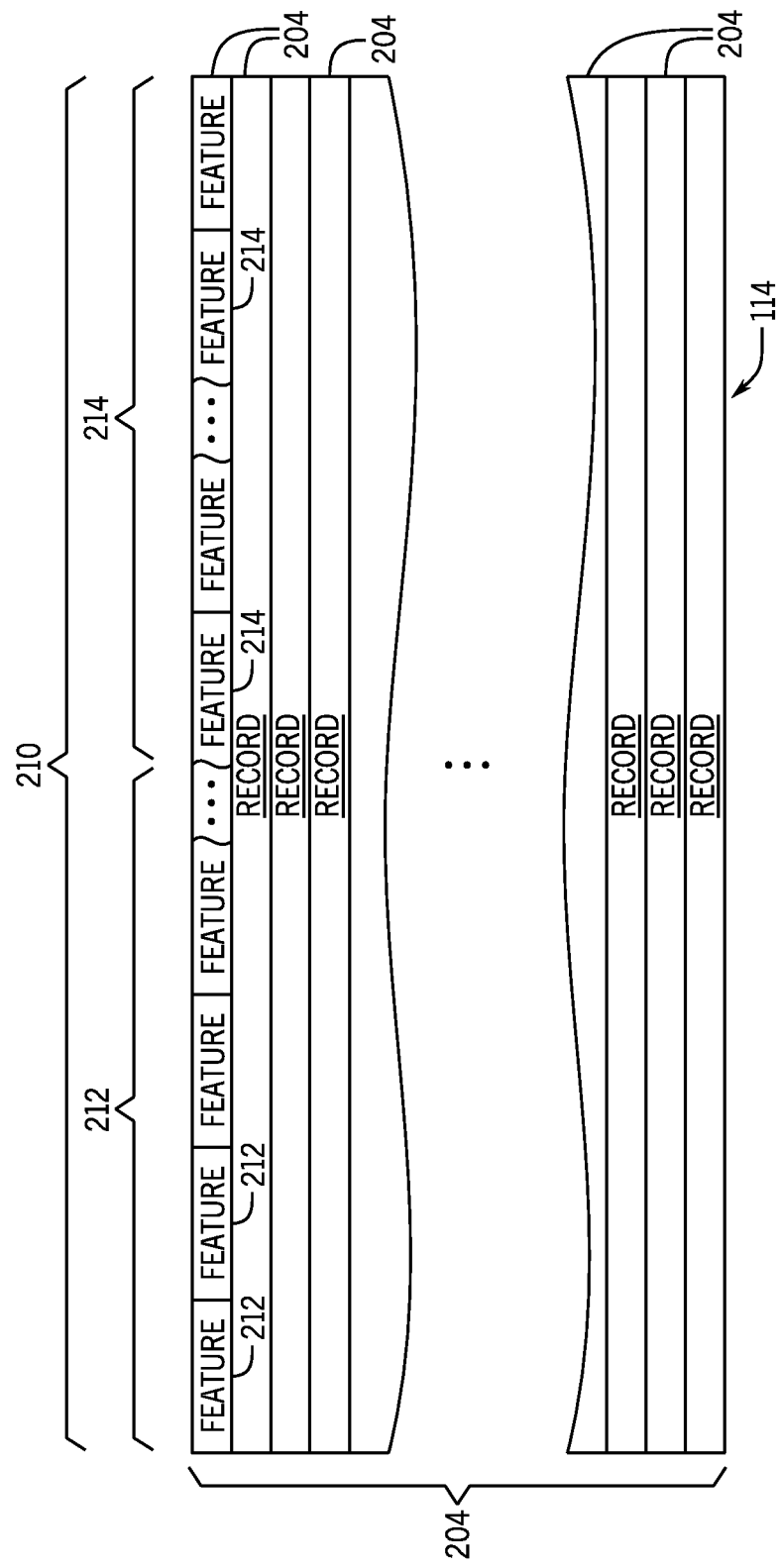


FIG. 2

4 / 9

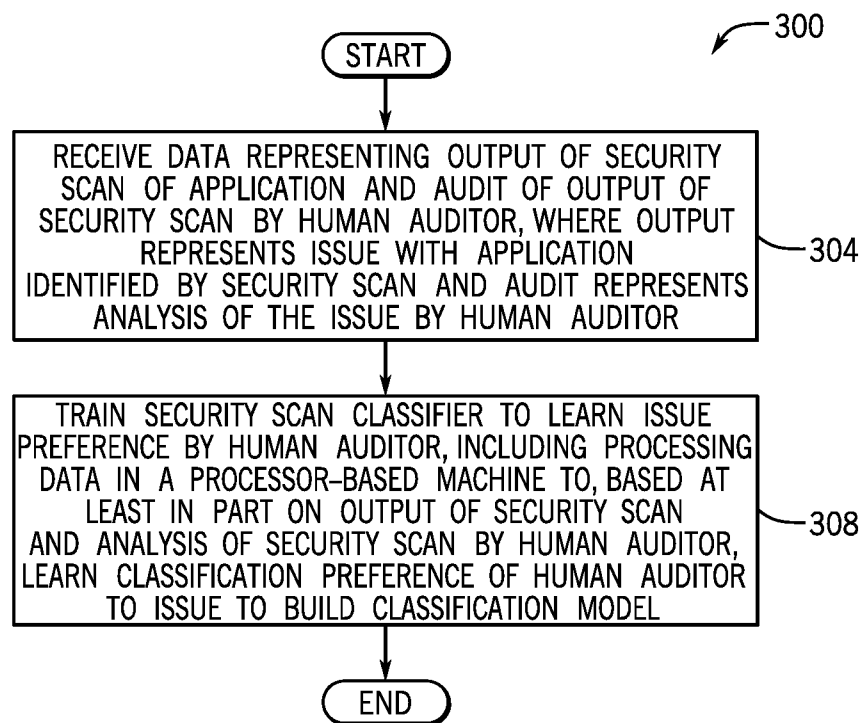


FIG. 3

5 / 9

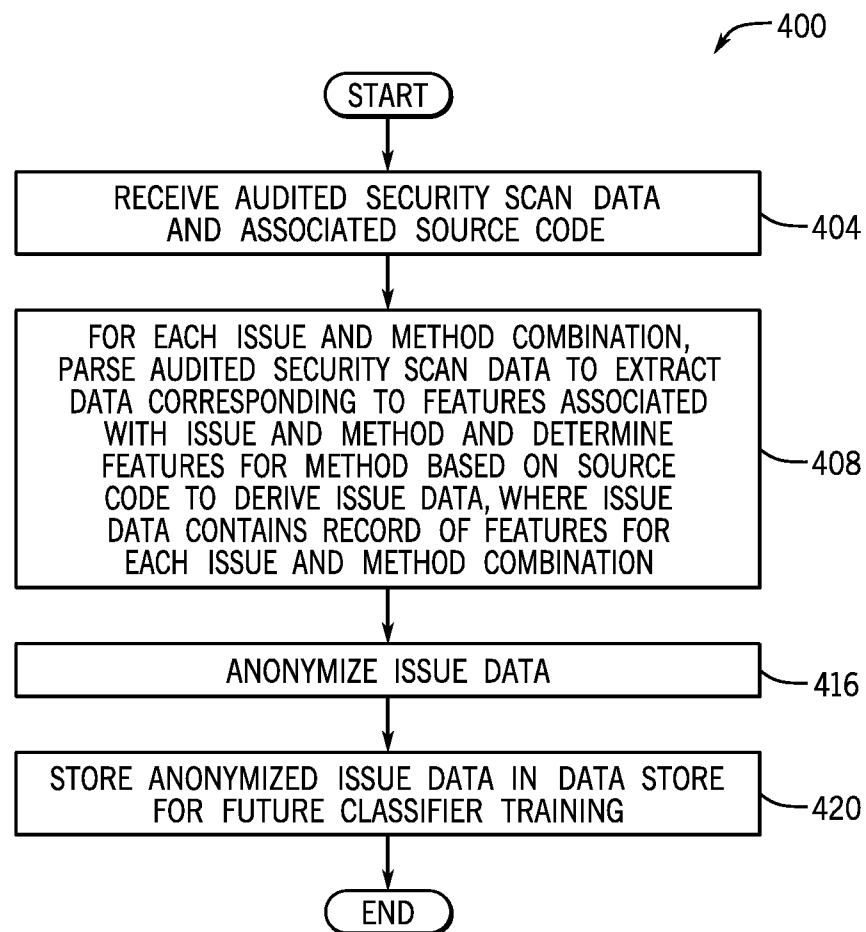


FIG. 4

6 / 9

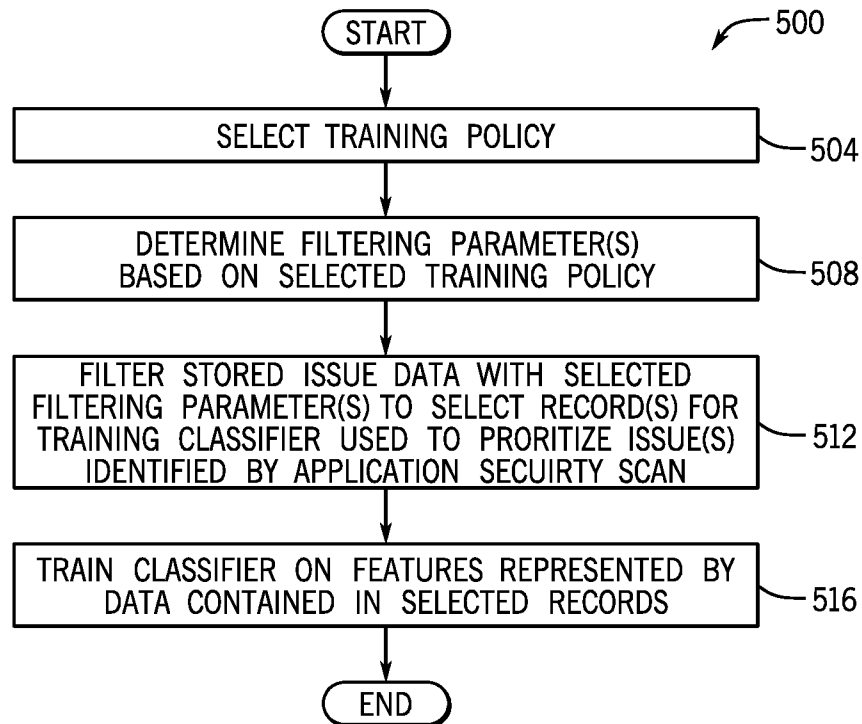


FIG. 5

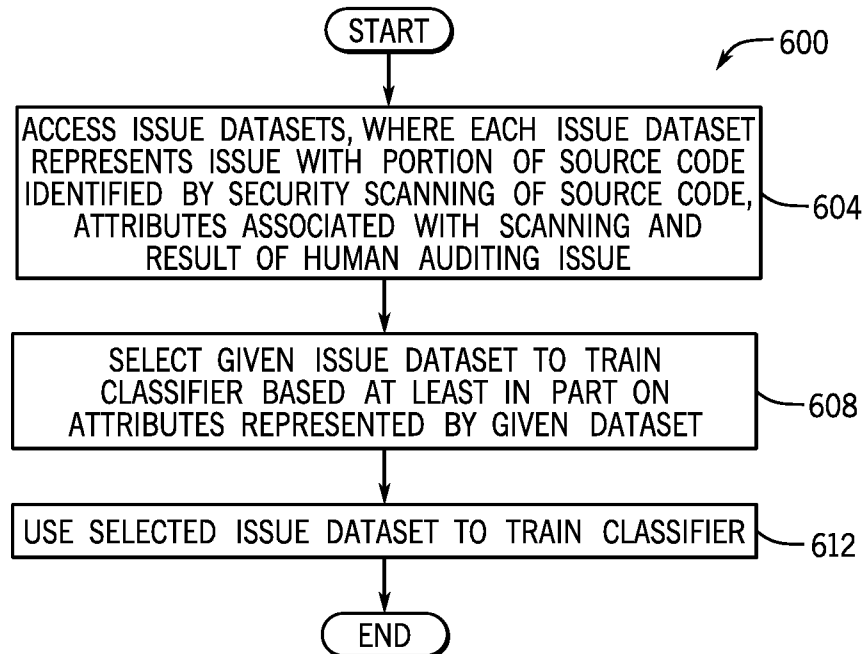


FIG. 6

7 / 9

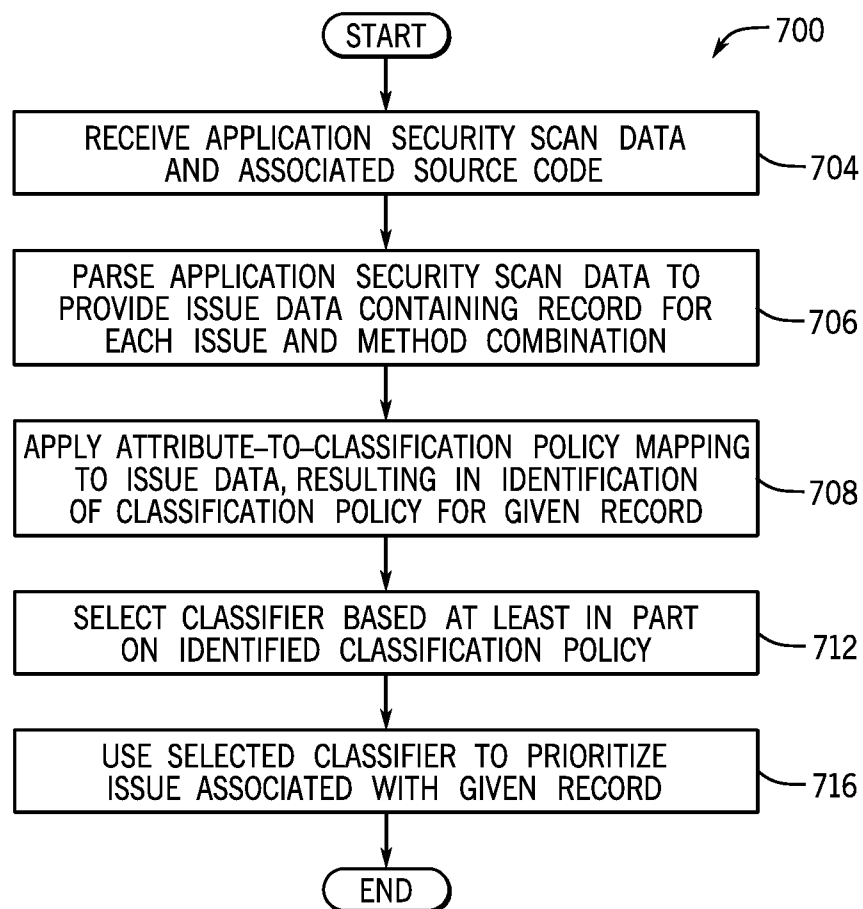


FIG. 7

8 / 9

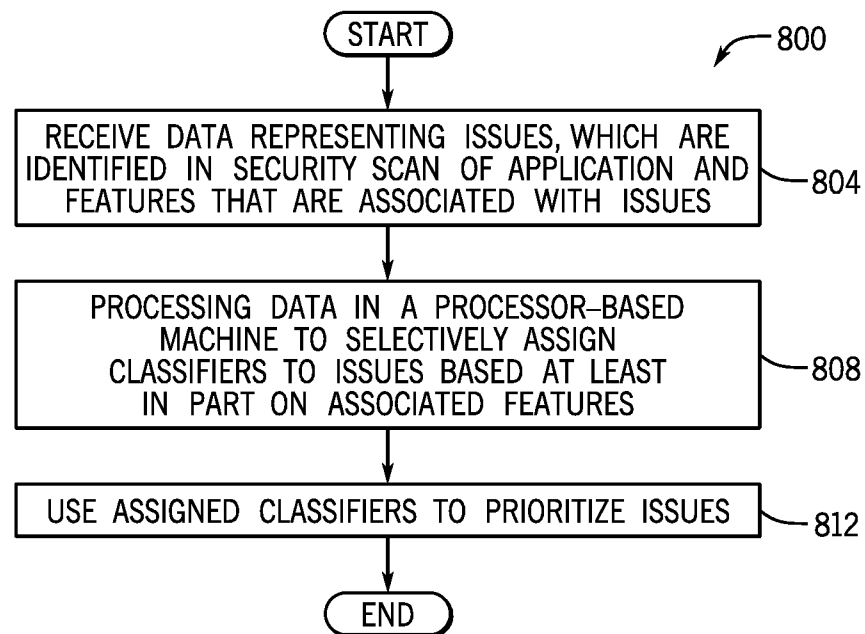


FIG. 8

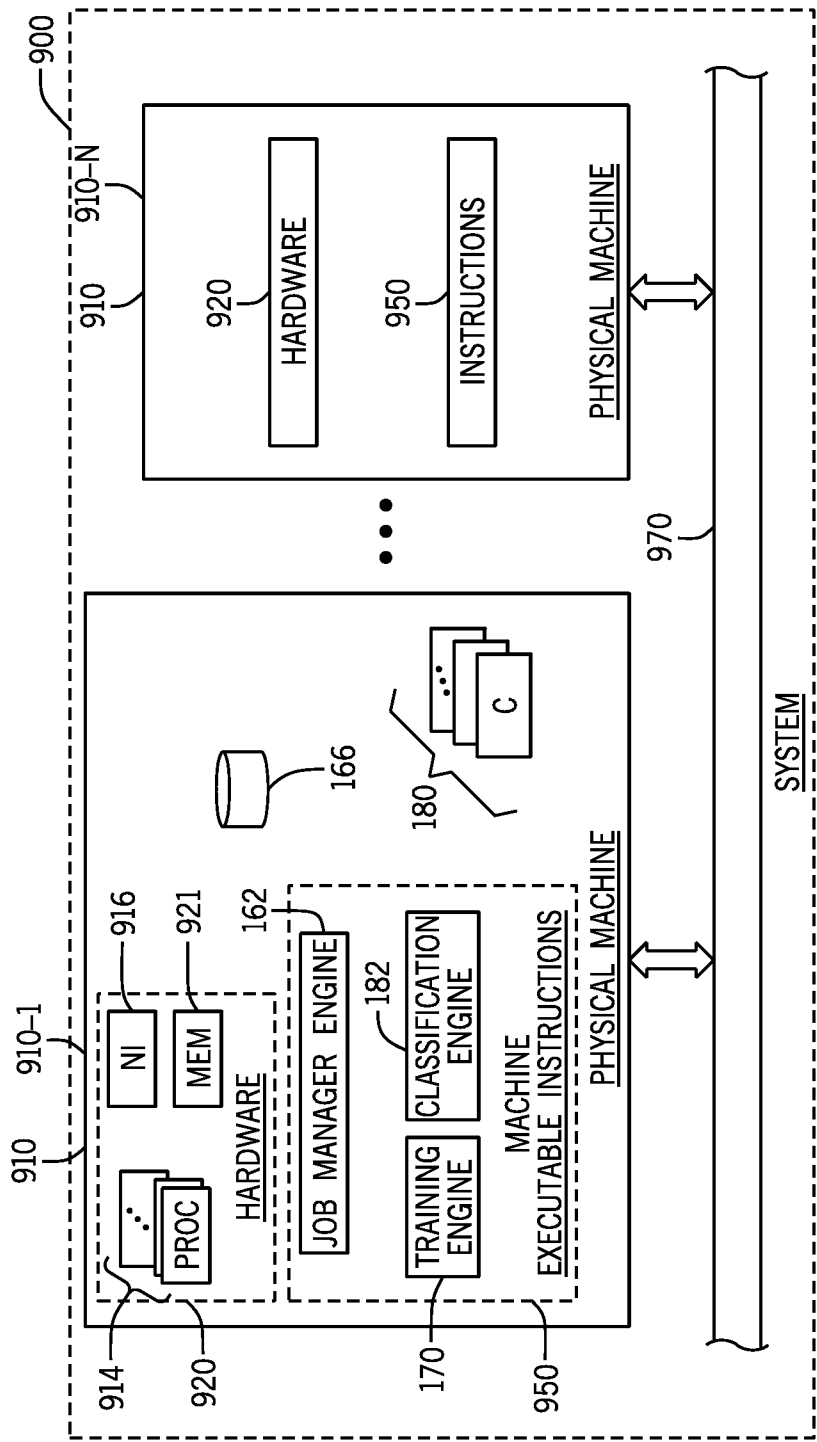


FIG. 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/044903**A. CLASSIFICATION OF SUBJECT MATTER****G06F 17/40(2006.01)i, G06F 21/00(2006.01)i**

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)

G06F 17/40; G06F 9/44; G06F 21/00; G06F 11/00

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

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

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

eKOMPASS(KIPO internal) & Keywords: application security, vulnerability, classifier, training, feature, attribute, and similar terms.

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2014-0090064 A1 (INTERNATIONAL BUSINESS MACHINES CORP) 27 March 2014 See paragraphs [0030]-[0031], [0040]-[0041], and [0043]; claim 1; and figures 1-2.	7-8
Y		1-6, 9-15
Y	US 2014-0075560 A1 (LOTEM GUY et al.) 13 March 2014 See paragraphs [0020] and [0034]; claim 1; and figure 2A.	1-6, 9-15
A	US 2015-0100940 A1 (AVAYA INC.) 09 April 2015 See paragraphs [0039]-[0041] and figure 4.	1-15
A	WO 02-062049 A2 (DODD, TIMOTHY DAVID et al.) 08 August 2002 See page 3, lines 2-13; page 7, line 30 - page 8, line 18; and figures 1, 3.	1-15
A	US 2008-0307525 A1 (MICHAEL D. NICKLE) 11 December 2008 See paragraphs [0032] and [0039]; and figure 3.	1-15



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

"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

25 May 2016 (25.05.2016)

Date of mailing of the international search report

25 May 2016 (25.05.2016)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea

Facsimile No. +82-42-481-8578

Authorized officer

NHO, Ji Myong

Telephone No. +82-42-481-8528



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2015/044903

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2014-0090064 A1	27/03/2014	US 8850589 B2 US 8918886 B2 WO 2014-052410 A2 WO 2014-052410 A3	30/09/2014 23/12/2014 03/04/2014 05/06/2014
US 2014-0075560 A1	13/03/2014	US 8806648 B2	12/08/2014
US 2015-0100940 A1	09/04/2015	US 9176729 B2	03/11/2015
WO 02-062049 A2	08/08/2002	AU 2002-243763 A1 AU 2002-244083 A1 US 2002-0104014 A1 US 2002-0147803 A1 US 2007-0250935 A1 US 7340776 B2 US 7712138 B2 WO 02-061544 A2 WO 02-061544 A3 WO 02-062049 A2	12/08/2002 12/08/2002 01/08/2002 10/10/2002 25/10/2007 04/03/2008 04/05/2010 08/08/2002 10/10/2002 08/08/2002
US 2008-0307525 A1	11/12/2008	None	