Mapping the Field of Software Security Metrics

Patrick Morrison, David Moye, and Laurie Williams
Department of Computer Science
North Carolina State University
Raleigh, NC USA
{pmorrison, cdmoye, lawilli3}@ncsu.edu

Abstract— **While security, or its absence, is a property of running software, many aspects of software requirements, design, implementation, and testing contribute to the presence or absence of security in the finished product. Assessing whether a given piece of software meets a set of security objectives is a multi-dimensional problem, and we do not yet have a clear picture of all of the dimensions. *The goal of this research is to support researcher and practitioner use of security measurement by cataloging available metrics, their validation, and the subjects they measure through conducting a systematic mapping study.* Our study began with 1,561 papers and narrowed down to 63 papers reporting on 346 metrics. For each metric, we identify the subject being measured, how the metric has been evaluated by researcher(s), and how the metric is being used. Approximately 85% of security-specific metrics have been proposed and evaluated solely by their authors. Approximately 40% of the metrics are not empirically evaluated, and many artifacts and processes remain unmeasured. Approximately 15% of the metrics focus on the early stages of development or on testing (1.5%). At present, despite the abundance of metrics found in the literature, those available give us an incomplete, disjointed, hazy view of software security.**

Categories and Subject Descriptors
D.2.8 [Software Engineering]: Metrics – *complexity measures, performance measures, process measures, product metrics.*

Index Terms—**Metrics, Measurement, Security**

I. INTRODUCTION

Software system stakeholders seek assurance that their interests, communications and data are secure. McGraw [1] defines *software security* as "engineering software so that it continues to function correctly under malicious attack." While security, or its absence, is a property of running software, many aspects of the engineering practices of software requirements, design, implementation, and testing contribute to the ultimate presence, or absence, of security in the finished product. Some fundamental security questions asked by development organizations can be answered with security metrics:

- Are we on track to release a secure product?
- How secure is the software I am considering including in my system?
- Does this system meet compliance obligations?

While we are not in a position to answer the posed questions directly, we can summarize the concerns that have been identified to date by examining the use of metrics for security during the software development lifecycle.

Providing useful metrics for the security of a software system is a difficult undertaking [2]. Pfleeger and Cunningham [3] consider dimensions ranging from the specification of systems to protocol verification to the psychology of software designers, users and attackers, suggesting that a range of metrics is needed to properly represent security for assessment and prediction.

Comprehensive approaches to providing security in software are likely to rely on comprehensive measurement of security in software. As shown by industrial schemes like Microsoft's Security Development Lifecycle [4], the SafeCode initiative[1], and the Cigital "*Building Security In* Maturity Model[2]", security must be addressed at every phase of software development. Researchers seek theories to explain security properties, and empirical validation of measurements of those properties. Both groups require an understanding of the available software security metrics.

The goal of this research is to support researcher and practitioner use of security measurement by cataloging available metrics, their evaluation, and the subjects they measure through summarizing the scholarly literature.

A summary of the state of the literature offers perspective on what has been accomplished, and what remains to be accomplished. We focus specifically on software, and the artifacts, processes and people involved in its development. For example, we do not consider measures of network, biometric, or cryptographic security.

According to Budgen [5], systematic mapping studies are "intended to 'map out' the research that has been undertaken rather than to answer a detailed research question." As a means of identifying the security properties to be measured, and the concerns involved in engineering secure software, we conduct a systematic mapping study of the metrics that have been applied to measuring the security of software during and after its development.

To assess the extent of the field of software security metrics and their evaluation and use, we pose the following research questions:

RQ1: What software security metrics have been proposed in the scholarly literature?

[1] http://www.safecode.org/
[2] http://bsimm.com/

RQ2: What is being measured by software security metrics?

RQ3: How are software security metrics evaluated in the literature?

RQ4: What phases of the software development lifecycle are measured by software security metrics?

Our initial search yielded a set of 1,561 papers. We narrowed the set to 63 papers that propose, evaluate and/or report on security metrics for software development. Our study provides context to software security researchers for evaluating existing and new security metrics for software development, and provides practitioners an inventory of security metrics for software development.

Our contributions include:

- A classification scheme for software development security metrics
- A summary of the metrics used to evaluate the security properties of software and its development
- Raw data related to software security metrics.

The remainder of this paper is organized as follows: Section II provides a glossary and background information on metrics. Section III presents related work. Section IV describes the methodology we followed in executing the mapping study. Section V provides our summarization of the data collected. Section VI reports on Limitations, and Section VII presents our discussion of the results.

II. BACKGROUND

To provide grounding for the topic of security metrics in software development, our mapping study, and our classification scheme, this section presents a glossary of metric-related terms, and literature on software metrics generally, and specifically on software security metrics.

A. Definitions

Attack – An intentional act by which an entity attempts to evade security services and violate the security policy of a system; A method or technique used in an assault [6].

Indicator: Any observable characteristic that correlates with a desired security property [7].

Measure: A way to ascertain or appraise value by comparing it to a norm; To apply a metric [8].

Measurement: The process by which numbers or symbols are assigned to attributes of subjects in the real world in such a way as to describe them according to clearly defined rules [9].

Metric: A quantitative measure of the degree to which a system, component, or process possesses a set attribute [10].

Security Measure: Assigns to each measured object a security indicator value from an ordinal scale according to well-defined measurement protocol [7].

Risk: The combination of the probability of an event and its consequence[3].

Security metric: A security measure and an associated set of rules for the interpretation of the measured data values [7].

Software security: We adopt McGraw's notion of "engineering software so that it continues to function correctly under malicious attack" [1]. For our notion of malicious attack, we also reference the IEEE definition of software security: "Protection of information and data so that unauthorized persons or systems cannot read or modify them and authorized persons or systems are not denied access to them."[3]

Software system: A software-intensive system for which software is the only component developed or modified[4].

Vulnerability: A flaw or weakness in a system's design, implementation, or operation and management that could be exploited to violate the system's security policy [6].

B. Software Metrics

Within the field of software engineering, few agree on the use of the words "metric" and "measure." For the purposes of this review, we are liberal in the definitions we accept from the literature. A good metric should be conceptually specific, quantitatively measurable, practically attainable, consistently measured without subjective criteria, and time-dependent [11]. However, even when metrics appear to be useful, difficulties arise attempting to validate metrics and determine their overall usefulness and practicality [12]. In addition, metrics are not valuable if the results of applying them cannot be understood effectively. Security metrics are no exception [13].

C. Software Security Metrics

What is a *security metric*? Jansen [8] quotes and discusses three variant definitions from the literature. To frame the text of this paper, we adopt Rudolph and Schwarz's [7] definition for a security metric "a security measure with an associated set of rules for the interpretation of the measured data values". Rudolph and Schwarz define a set of attributes for describing security metrics, including the "Target" the metric measures, the lifecycle "Phase" when the metric is measured, and whether the metric is prescriptive or goal-oriented. *Prescriptive* metrics characterize the quality with which a process step within the development lifecycle is performed. *Goal-oriented* metrics measure the quality of the product.

We supplement the Rudolph and Schwartz framework with notions taken from Savola's [14] security metrics taxonomy, which characterizes security metric properties and applications. At a high level, Savola provides three categories of metrics:

[3] ISO/IEC 16086 (IEEE Std 16085-2006) - Systems and Software Engineering - Life Cycle Processes - Risk Management

[4] IEEE Std 1362-1998 - IEEE Guide for Information Technology - System Definition - Concept of Operations (ConOps) Document

- Organizational metrics describe attributes of organizational programs and processes.
- Technical metrics describe software artifacts, e.g., requirements, specifications, designs, code.
- Operational metrics describe running systems and their environments.

III. Related Work

Verendel [15] presents a survey focused on measuring operational security, addressing the ability to "function correctly under malicious attack." Our mapping study additionally considers the engineering of secure software, seeking measurements of the process, tools, people, and the software produced. Rudolph and Schwarz [7] surveyed scholarly articles on "security indicators", where an indicator is defined as "an observable characteristic that correlates with a desired security property." In addition to what Rudolph and Schwarz studied, we seek to characterize the subjects being measured. Meneely, Smith, and Williams [18] review metric validation, and suggest a scheme for choosing validation criteria. We only consider validation in terms of the high-level approach chosen by the researchers, e.g., User Study, Theoretical, or Opinion.

IV. METHODOLOGY

We subdivide how we approach the mapping study into four components: our search strategy for identifying papers, our selection criteria for including papers, our classification scheme for collecting data on each metric, and our procedure for extracting metric information from each paper.

A. Search Strategy

In this section, we lay out the process used to conduct our systematic mapping study.

1) Databases

We based our selection of online databases on the most common databases used in Software Engineering Systematic Literature Reviews (SLRs), and in Systematic Mapping Studies (SMSs), and on sources used in previous software security metric literature reviews [7], [15]. The data sources in this study include online databases, conference proceedings, and academic journals. The list is as follows: ACM Digital Library, IEEE Xplore, and Elsevier.

2) Search terms and strategy

For each research question, we selected the terms the first and second authors agreed on as most relevant.

Table 1. Research Question Keywords

Research Question	Keyword
RQ1	"software", "security", "measure", "metric"
RQ2	RQ1 terms
RQ3	RQ1 terms + "validate"
RQ4	RQ1 terms

Table 2. Keyword Synonyms

Keyword	Synonym(s)
software	"application"
security	"vulnerability"
measure	"measurement", "measure", "indicator", "attribute", "property"
metric	"measurement", "measure", "indicator", "attribute", "property"
validate	"validation", "evaluate", "evaluating", "evaluation", "quantify", "quantifying", "quantification", "quantified", "quantitative", "assess", "assessment", "measure", "measurement", "appraisal", "analyze", "measuring", "analysis", "assurance", "scoring"

For each term associated with a research question, we identified synonyms in the titles and abstracts of previous surveys [7], [15] of security measures and metrics.

The organization of the research questions is that RQ2, RQ3, and RQ4 are subsets of RQ1. We created database search phrases based on RQ1 and collected studies based on those search phrases, filtering from the set collected for RQ1 to answer the other research questions.

Our base search phrase is:

(security OR vulnerability) AND (metric OR measure OR measurement OR indicator OR attribute OR property)

3) Search Process Evaluation

We followed Zhang [16] in evaluating the quality of our search results. An ideal search process would return all relevant papers (sensitivity = 1.00), and only relevant papers (precision = 1.0). Such a set of papers would be a "gold standard." In practice, we do not know the set of relevant papers in advance, so we must estimate, using what Zhang terms a "quasi-gold standard" (QGS) which is a set of relevant papers in the literature, chosen prior to the search process. The QGS is used as a measure of how well each search string locates relevant papers in the absence of a gold standard. Zhang [16] defines sensitivity as the ratio of the number of returned papers retrieved to the total number of relevant studies. Quasi-sensitivity (QS) is the number of papers returned by a search to the number of returned papers that are present in the QGS. QS estimates how well the search string locates relevant papers within the searched corpus. By measuring the performance of the search in returning members of the QGS from a given search engine, compared with the number of QGS papers in the search engine, an estimate can be made of search performance. For example, if there are 10 of the QGS papers in the ACM library, and the search string returns 8 of them, QS would be 0.8.

The first and second authors identified a set of 39 software development security metrics papers, developed by reviewed the papers collected in the previous security metrics literature reviews [7], [15] Each author read the titles and abstracts of the collected papers independently,

and made a list of candidates. The authors then discussed each list, applying the following selection procedure:

1. If both authors agreed the paper described a software development security metric, it was included in the final QGS list.
2. If both authors agreed the paper did not describe a software development security metric, it was excluded from the final QGS list.
3. Disagreement was discussed. If agreement could be reached, the appropriate action listed above was taken. If agreement could not be reached, the paper was included in the final list.

The results of each database search were compared with the QGS set of papers. The quasi-sensitivity for each database search is reported in Table 3.

B. Selection Criteria

We developed a list of criteria to assess whether the papers found during the search process met our objectives.

1) Inclusion Criteria

- Primarily related to measuring software security in the software development process and/or its artifacts, for example software artifacts (e.g., source code files, binaries), software process (e.g., requirements phase, design, implementation, testing), and/or software process artifacts (e.g., design and functional specifications)
- Measurements and/or metrics are main subject
- Refereed paper
- Published since 2000

2) Exclusion Criteria

- Sources related to sensors
- Sources related to identity, anonymity, privacy
- Sources related to forgery and/or biometrics
- Sources related to network security (or vehicles)
- Sources related to encryption
- Sources limited to database security
- Sources related to imagery, audio, or video
- Sources specific to single programming languages

3) Study Quality Assessment

We developed a Quality Assessment Checklist for whether to include each paper, as follows:

1. Is a primary or secondary goal of the paper to describe, define, or evaluate a metric or measurement of software security?
2. Does the paper align with our inclusion criteria?
3. Is the paper peer-reviewed?

We also established a scoring procedure for resolving differences between raters when disagreement was found:

4. Scoring
   a. Question Scoring Scale: No: 0, Partial: 0.5, Yes: 1, Two raters.
   b. Complete agreement
      i. "Yes" from both raters: paper is selected
      ii. "No" from both raters: paper is rejected
   c. Partial agreement – combinations between 0 and 2. Raters discuss, find agreement, or agree to disagree.
      i. Agreement processed according to the rules for complete agreement
      ii. Papers are selected in the presence of unresolved disagreement.

C. Metric Classification Scheme

To support answering our research questions, we developed a set of data elements to be collected for each metric. We began with the Rudolph and Schwartz survey [7] classification scheme (R&S indicates the element was defined in their framework) and the Savola security metrics taxonomy [14], adapting the elements and their values to our goals and questions, and adding new elements where the original schemes did not address our questions.

Elements linked to RQ1:

- Metric name: the "given name" of the metric defined or used in the paper. (R&S)
- Metric Category: The category values were synthesized extraction to summarize metric usages.

Elements linked to RQ2:

- Type: what kind of attribute is measured? (R&S)
  - Prescriptive: Measures the quality with which a process step within the development lifecycle is performed
  - Goal-oriented: Measure the quality of the product itself
- Target: class of subject measured by the metric (R&S).
  - Product: Refers to the security of the target, e.g., software products or parts of them source code, components, systems, etc.
  - Process: Refers to security-related parts of a process, e.g., a development process, or a maintenance process.
  - Resources: Refers to security-related attributes of resources used by a product or process.
- Subject measured: Synthesized and named based on assessment of extractor: "Source Code", "Project", and "Component" are examples.
- Phase (of the development lifecycle, R&S): Requirements, Design, Implementation, Testing, Deployment, Operations
- Unit of measurement: denotes type of measurement scale (e.g., Nominal, Ordinal, Interval, Ratio), or specific instance of measurement scale(e.g., Count, Probability, Duration, Rate, Currency, Percentage)

Elements linked to RQ3:

- Evaluation technique: Opinion, Theoretical, Academic user study, Industry user study, Reports from production.

- Affect identifies how the metric value is measured, where "Quantitative" indicates objective, systematic, empirical measurements and "Qualitative" indicates subjective measurements based upon observation.

Elements linked to RQ4:

- Phase (of the development lifecycle, R&S): Requirements, Design, Implementation, Testing, Deployment, Operations

We also collected demographic and audit information for each metric. For demographic purposes, we assigned unique paper and metric numbers to identify each metric and its usage across papers, and a set of sub-categories to track distinctions not made by our categorization scheme (e.g., we treat coupling and cohesion as part of the Dependency category), For audit purposes, we tracked the name of the extractor and auditor for each row, as well as the extraction and audit dates.

D. Data Extraction

The metric extractor (in the present case, the first or second authors) reads a paper, identifies each metric defined or used in the paper, and collects the data for each element in the metric classification scheme.

The first author applied the extraction procedure described in the classification guide to every paper in the final set of 63 papers. The second author applied the extraction procedure to a sample of studies. We then discussed and resolved differences in classification where they were present.

V. RESULTS

This section presents our search results, and summaries of the data extracted from the selected papers. Based on the data extraction procedure, we tabulated the metrics, subjects, scales, evaluation means, and uses. The tabulated data provides an overview of "evidence clusters" and "evidence deserts" for software security in the software development process as well as detailed information about each metric. The results in Section V.B provide summary answers to the paper's research questions based on the excerpts from the tabulated data. The full data set is presented in Appendices B, C, and D.

A. Search Results

The search phrases returned 1,561 papers. Of these, 173 papers were duplicates among two or more sources. We measured inter-rater agreement using Cohen's Kappa [17], obtaining 0.93 for the pass over titles, and 0.81 for the pass over the abstracts. Kappa scores over 0.8 indicate relatively strong agreement. Applying our inclusion and exclusion criteria through unanimous agreement of two raters' evaluations of document titles resulted in a set of 332 papers.

The papers were then evaluated by reading the abstracts. The abstracts were then compared against the exclusion and inclusion criteria, reducing the source set to 185 papers, which were evaluated based upon their full text.

TABLE 3. SEARCH RESULTS

Database	Initial Search Results	QS	Precision
ACM Digital Library	223	0.35	0.031
IEEE Xplore	502	0.58	0.014
Elsevier	836	1.0	0.012

The full text for the remaining papers was inspected for basic compliance by giving each article a 10-minute review, applying the Study Quality Assessment criteria, yielding 78 papers, then a full-text analysis resulting in 63 included in the final survey. We applied the two rater selection procedure described in Section III.1.3 for each pass over the set of papers.

B. Data Extraction Results

Applying the data extraction scheme described in Section IV.D, we identified 346 metrics, shown in Appendix A. We now address each research question from the perspective of the data collected.

1) RQ1: What software security metrics have been proposed in the scholarly literature?

We identified 346 unique metrics across the 63 papers selected. Space does not permit a discussion of each metric, but several themes emerged. We have arranged these themes as ten metric categories. Our category titles, listed in Appendix A, were initially seeded from the papers themselves, listed in Appendix B. For example, Complexity (P25, P32, P34), Dependency (P25, P43), and Churn (P25) are used by their citing papers to summarize lists of metrics denoting each concept. For these cases, we identified a metric as belonging to the Complexity category when the containing paper did so, or based on past references to the metric in the literature. In many cases, papers compute variants of a basic metric, e.g., "Complexity" by averaging ("Average Complexity"), totaling ("Total Complexity"), or finding the maximum ("Maximum Complexity") or minimum ("Minimum Complexity"). We place these variants in the base metric's category. We followed a similar approach for each of the ten categories. We defined ten categories to summarize the kinds of metrics identified (metric count follows category name in parenthesis), listed below:

- Churn (11): The amount of change in a measured subject, typically generated by counting the additions, deletions and changes made.
- Complexity (22): The difficulty with which a subject is created, understood, and/or tested.
- Cost (5): The expense incurred in experiencing or responding to a security-related loss
- Coverage (100): The ratio (percentage) of sub-parts of a subject to which some attribute applies, compared to the total number of sub-parts within that subject.
- Dependency (33): The level of interconnection between subjects

- Effort (23): The difficulty of attacking or defending some subject
- Organization (18): Measures of the team and/or organization
- Size (50): Measures of the size of the subject
- Strength (30): Positive measures of an subject's security properties
- Weakness (55): Negative measures of an subject's security properties

a) Traditional Software Metrics: Churn, Complexity, Dependency, Size

Many of the metrics identified across the categories of Churn, Complexity, Dependency, and Size were first defined and used in the broader software metrics literature. Application of these metrics to evaluating and predicting security properties is an extension of their previous use in defect prediction (e.g., [19]). Each of the categories will now be discussed.

CHURN (TRADITIONAL)

Typical measures of source code churn sum the number of lines added, deleted, or changed during some unit of time. In one case, (P53), only the number of lines deleted is tracked. "Relative Churn" (M271, P63) and "Percentage Interactive Churn" (M218, P63) normalize the size of the churn to the size of the changed file and consider overlapping changes between developers, respectively.

COMPLEXITY (TRADITIONAL)

Variants of McCabe's Cyclomatic Complexity were the most commonly used metrics (20 references, nine papers). We summarized these as Complexity (M50). Fan In and Fan Out, measures of how connected a given code object is to other code objects, were also common (11 references, four papers). Beyond common complexity measures, "Average Service Depth" (ASD) (P45), measures the complexity of a service provided to users by tracking the number of constituent services used in providing the given service and relating ASD to the "attackability" of services.

COST (SECURITY-SPECIFIC)

Five papers consider the cost of a security breach or the risk of a breach in financial terms, most often in terms of the actual expenses incurred in recovering from a breach.

COVERAGE (BOTH)

In software engineering, "Coverage" typically refers to the percentage of the total code base that is executed when a test suite is run. Our notion of Coverage abstracts out the idea of computing the ratio of subjects (e.g., lines of code (LOC), number of classes or files) possessing some attribute (test coverage) to the total number of subjects (e.g., LOC, classes, or files). We observed this more general idea of coverage in 100 metrics, nearly 1/4 of all the metrics. P1 defines notions of "Classified" and "Critical" to describe software access to information, and defines a set of metrics describing the coverage of "Classified" and "Critical" classes, attributes related to the complete set of classes and attributes in the software. P23 applies coverage to measures of security-related requirements, design decisions, test cases, and flaws identified. Several papers considered coverage of operational aspects of software, e.g., logging (P4, P28) and aspects of session management (P2, P62).

The most security-specific categories measure aspects of how some subject fulfills or fails to fulfill one or more security properties. We have divided these into negative measures, termed "Weakness," and positive measures, termed "Strength."

DEPENDENCY (TRADITIONAL)

Beyond measures of size and complexity of individual software functions, measures of dependency characterize how software subjects connect with each other. The core concepts for dependency metrics are drawn from graph theory, and applied to software by treating software subjects (functions/methods, classes, objects, services) as nodes in the graph and relationships as edges in the graph. Various metrics, for example, "in degree", "out degree", and "betweenness" (P32) are computed to represent software attributes. There were several papers that measured Coupling (P1, P31, P42, P45, P48, P51, P61) and/or Cohesion (P61), internal software metrics that measure software relationships. We treated each of these as a form of dependency for purposes of our categorization.

EFFORT (SECURITY-SPECIFIC)

We identified notions of attacker, defender, and developer effort. The Common Vulnerability Scoring System (CVSS) (P9, P16, P17, P24, P60, CVSS[5]) provides ordinal classifications for indicating the difficulty an attacker has in reaching and exploiting a vulnerability along the dimensions of how the vulnerability is reached ("Access Vector"), whether an account compromise is required ("Authentication") and how sophisticated an attack is required ("Access Complexity"). CVSS's notions of "Confidentiality Requirement", "Integrity Requirement", and "Availability Requirement" offer similar metrics for indicating developer and defender effort required to achieve various security properties. One paper (P19) defines metrics for evaluating the effort put in to the design and inspection of security mechanisms.

ORGANIZATION (SECURITY-SPECIFIC)

Four papers considered measures of the size and nature of the team and organization developing the software. Three papers used the number of developers to measure the notion of whether "too many cooks spoil the broth" from a security perspective. In P25 and P63, Meneely develops and uses measures of how developer interactions and networks affect the security properties of software.

[5] http://www.first.org/cvss/cvss-guide.html

SIZE (TRADITIONAL)

The most basic software measurement, Lines of Code (LOC), is also among the most used metrics in the security metrics literature (13 references, 9 papers). Other tracked source code elements include variable declarations and number of functions (P32), blank and comment lines (P43), and counts of variables and classes (P48, P61). One paper (P10) advocates counting more specific source code-level attributes, e.g., arithmetic expressions and array indices, as these attributes lie beneath buffer overflows. Extending the notion of defining code attributes, another paper (P1) defines notions of "Classified" data attributes in code and "Critical" classes that reference Classified data attributes. The most basic metrics in these papers count the number of these attributes present, making them "Size" metrics. Several papers track size metrics for attributes of software design (P23) and requirements (P52). Several papers tracked counts of attacks (P29) and vulnerabilities (P58).

"Attack Surface Metric" (ASM) measures software size from an attacker's point of view (P13). ASM counts the number of resources accessible to a user of the software. Studies have shown correlations between ASM and security challenges (P7, P13, P37, P38).

STRENGTH (SECURITY-SPECIFIC)

Metrics of software security strength (positive performance on security properties), like those of weakness, range across the development process, from requirements and design, to development, to operations. One paper (P22) attempts to operationalize measures of several security design principles, e.g., "Least Privilege", "Compartmentalization", and "Defense-in-Depth." Subjective measures include CVSS's "Report Confidence" (P17), "Independence of Verification" (P19), "risk control", "software project management experience" and "Trustworthiness" (P46). Operational measures include "Vulnerability Free Days" (P18) and "Mean Time to Failure" (P45).

WEAKNESS (SECURITY-SPECIFIC)

Metrics of software security weakness (negative performance on security properties) range from subjective estimates of project, team, and process risk (P46, P56, P60) to measures of development (P23, P42) and operational weaknesses (P4) to measures of vulnerabilities found in the software subject (P2, P21, P23, P28, P42, P58). The CVSS "Confidentiality Impact", "Integrity Impact" and "Availability Impact" scores, each indicting the relative seriousness of vulnerability's impact on the software system, are representative of weakness metrics.

2) RQ2: What is being measured by software security metrics?

The metrics identified in this Mapping Study were categorized in three dimensions: Subject, Lifecycle Phase, and Target.

We identified 13 distinct subjects (metric counts follow names): Source Code (175), System (123), Component (52), Binary (29), Software version (28), Organization (12), Misuse case (8), Requirements (8), Commit (7), Project (6), Design (4), Security mechanism (4), Network (3), Component inspection (2), User (1).

We note that measures of the source code are most frequent, often reflecting the application of traditional software metrics to measuring security. Measures of running systems (System, Software Version) (151) comprise the next largest collection of metrics. Measures of the development process and its non-source code artifacts are relatively few.

In the Target dimension, Product (383), Process (67), and Resources (9), metrics that measure the actual Product dominate both those that measure the Process and those that measure Resources.

3) RQ3: How are software security metrics evaluated in the literature?

Broken down by Evaluation technique, the counts were: Industry User Study (121, 35%), Academic User Study (85, 25%), Opinion (61, 18%), Theoretical (51, 15%), Reports from production (18, 5%), Not Described (10, 3%). Opinion and Theoretical evaluations make up 33% of the evaluation techniques.

Related to the evaluation of the metrics, is Affect, denoting whether a metric is Quantitative (310, 67%) or Qualitative (68, 15%) (86, 18% were unidentified).

4) RQ4: What phases of the software development lifecycle are measured by software security metrics??

The most common lifecycle phase of development for metrics is Implementation (229), whereas the least common phase is Testing (7) (followed closely by Requirements, and Design). The Implementation phase is dominated by Size and Coverage metrics; and has relatively few metrics for Cost, Effort, and Strength. The Implementation phase is also the only phase that has metrics representing all of the 10 categories used in this study.

VI.LIMITATIONS

If we have seeded our Quasi-Gold-Standard (QGS) with the wrong papers, we may have excluded relevant papers

TABLE 4. LIFECYCLE PHASE BY METRIC CATEGORY

	Rqt's	Design	Impl.	Test	Ops	Tot.
Churn			15			15
Complexity		7	41	1		49
Dependency		12	22		1	35
Size	5	6	48		9	68
Organization			22			22
Cost	1		1		4	6
Effort	4	1	5		29	39
Coverage	14	3	54	3	42	116
Weakness	2	2	10	1	43	58
Strength	2	6	6	2	20	36
Total	28	37	224	7	148	444

from our results. We drew our results from three search engines, ACM, IEEE, and Elsevier, limiting our selection of metrics papers to what is available in their indexes.

Our QS scores were low for ACM and IEEE, suggesting that we may have missed relevant papers. While we attempted to be comprehensive in our search strings and result parsing, our approach may have missed papers. Limiting our search to the scholarly literature excluded existing standards as well as industry experience reports disseminated by other means. Software development organizations may choose not to report whether they are using metrics, limiting our observations to discussion of the scholarly literature. Our metric classification scheme reflects our own biases in the data elements selected for each metric, and both the scheme and the biases of each author affect the values selected for each data element for each metric. We attempted to reduce bias by applying our two rater scheme, as well as more informal discussions among the authors.

Drawing inferences from the fields we classified depends on how accurately our choices match objective reality. We did not attempt a second approach, or a second set of extractors, to compare results, so our measures of validity are weak. Data elements we synthesized (Category, Measured Subject) are especially subject to this limitation, though we had two extractors check each metric-category assignment.

VII.DISCUSSION

We reported on the results associated with each research question in section V. Here, we offer several further observations. At first glance, 346 software security metrics appears to be an abundance of metrics, enough for measuring any conceivable application. On closer examination, we identified opportunities for new metrics and evaluation of existing metrics. The most common 'security metrics' are traditional software metrics, e.g. Complexity, Churn and Lines of Code, applied to measuring security. After traditional metrics, metrics that characterize vulnerabilities, e.g. CVSS, are well represented. Attempts, like P22, to define metrics measuring how well code or components supports a security principle, e.g. 'Least Privilege' are a valuable contribution. A majority (60%) of metrics in the surveyed literature are evaluated through either Industry (35%) or Academic Studies (25%), but note that the same figures show that 60% of metrics have not been applied in an industrial setting, calling for further empirical studies.

Most (85%) security-specific metrics have been proposed and evaluated solely by their authors. Few metrics (~15%) focus on the early stages of development or on testing (1.5%). Applying ideas from how implementation concerns are measured, e.g. metrics for assessing designs and test suites, and metrics for assessing team strength, are one possible research direction. . Many metrics (~40%) are not empirically evaluated, and many artifacts and processes remain unmeasured. Following through on the evaluation and use of proposed metrics is a natural research direction. At present, despite the abundance of metrics found in the literature, those available give us an incomplete, disjointed, hazy view of software security.

ACKNOWLEDGMENTS

Thanks to the RealSearch group for much helpful feedback during the development of this paper.

REFERENCES

[1] G. McGraw, *Software Security: Building Security In*. Addison-Wesley Professional, 2006.

[2] "Four Grand Challenges in Trustworthy Computing." Computing Research Association, Warrenton, VA, 2003.

[3] S. L. Pfleeger and R. K. Cunningham, "Why Measuring Security Is Hard," *Secur. Priv. IEEE*, vol. 8, no. 4, pp. 46–54, Jul. 2010.

[4] M. Howard and S. Lipner, *The Security Development Lifecycle*. Redmond, WA, USA: Microsoft Press, 2006.

[5] D. Budgen, M. Turner, P. Brereton, and B. Kitchenham, "Using mapping studies in software engineering," in *Proceedings of PPIG*, 2008, vol. 8, pp. 195–204.

[6] R. Shirey, *Internet Security Glossary, Version 2*. IETF, 2007.

[7] M. Rudolph and R. Schwarz, "A Critical Survey of Security Indicator Approaches," in *Proceedings of the 2012 Seventh International Conference on Availability, Reliability and Security*, Washington, DC, USA, 2012, pp. 291–300.

[8] W. Jansen, W. Jansen, P. D. Gallagher, and D. Director, *Directions in Security Metrics Research*. 2009.

[9] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*, 2nd ed. Boston, MA, USA: PWS Publishing Co., 1998.

[10] "IEEE Standard Glossary of Software Engineering Terminology," *IEEE Std 61012-1990*, pp. 1–84, Dec. 1990.

[11] N. Pham, L. Baud, P. Bellot, and M. Riguidel, "A Near Real-Time System for Security Assurance Assessment," in *Proceedings of the 2008 The Third International Conference on Internet Monitoring and Protection*, Washington, DC, USA, 2008, pp. 152–160.

[12] N. F. Schneidewind, "Methodology for Validating Software Metrics," *IEEE Trans Softw Eng*, vol. 18, no. 5, pp. 410–422, May 1992.

[13] M. Ouedraogo, D. Khadraoui, H. Mouratidis, and E. Dubois, "Appraisal and Reporting of Security Assurance at Operational Systems Level," *J Syst Softw*, vol. 85, no. 1, pp. 193–208, Jan. 2012.

[14] R. Savola, "Towards a Security Metrics Taxonomy for the Information and Communication Technology Industry," in *Proceedings of the International Conference on Software Engineering Advances*, Washington, DC, USA, 2007, p. 60–.

[15] V. Verendel, "Quantified security is a weak hypothesis: a critical survey of results and assumptions," in *Proceedings of the 2009 workshop on New security paradigms workshop*, New York, NY, USA, 2009, pp. 37–50.

[16] H. Zhang, M. A. Babar, and P. Tell, "Identifying Relevant Studies in Software Engineering," *Inf Softw Technol*, vol. 53, no. 6, pp. 625–637, Jun. 2011.

[17] J. Cohen, "A Coefficient of Agreement for Nominal Scales," *Educ. Psychol. Meas.*, vol. 20, no. 1, pp. 37–46, 1960.

[18] M. D'Ambros, M. Lanza, and R. Robbes, "Evaluating Defect Prediction Approaches: A Benchmark and an Extensive

Comparison," *Empir. Softw Engg*, vol. 17, no. 4–5, pp. 531–577, Aug. 2012.

[19] B. Alshammari, C. Fidge, and D. Corney, "A Hierarchical Security Assessment Model for Object-Oriented Programs," *Qual. Softw. Int. Conf. On*, vol. 0, pp. 218–227, 2011.

[20] E. A. Nichols and G. Peterson, "A Metrics Framework to Drive Application Security Improvement," *IEEE Secur. Priv.*, vol. 5, no. 2, pp. 88–91, 2007.

[21] K. Hajdarevic and P. Allen, "A new method for the identification of proactive information security management system metrics," in *Information Communication Technology Electronics Microelectronics (MIPRO), 2013 36th International Convention on*, 2013, pp. 1121–1126.

[22] X. Cheng, N. He, and M. S. Hsiao, "A New Security Sensitivity Measurement for Software Variables," in *Technologies for Homeland Security, 2008 IEEE Conference on*, 2008, pp. 593–598.

[23] J. Pamula, S. Jajodia, P. Ammann, and V. Swarup, "A Weakest-adversary Security Metric for Network Configuration Security Analysis," in *Proceedings of the 2Nd ACM Workshop on Quality of Protection*, New York, NY, USA, 2006, pp. 31–38.

[24] S. Nasiri, R. Azmi, and R. Khalaj, "Adaptive and quantitative comparison of J2EE vs. .NET based on attack surface metric," in *Telecommunications (IST), 2010 5th International Symposium on*, 2010, pp. 199–205.

[25] G. Schudel and B. Wood, "Adversary Work Factor As a Metric for Information Assurance," in *Proceedings of the 2000 Workshop on New Security Paradigms*, New York, NY, USA, 2000, pp. 23–30.

[26] K. Scarfone and P. Mell, "An analysis of CVSS version 2 vulnerability scoring," *2013 ACM IEEE Int. Symp. Empir. Softw. Eng. Meas.*, vol. 0, pp. 516–525, 2009.

[27] S.-T. Lai, "An Analyzer-Based Software Security Measurement Model for Enhancing Software System Security," in *Software Engineering (WCSE), 2010 Second World Congress on*, 2010, vol. 2, pp. 93–96.

[28] X. Song, M. Stinson, R. Lee, and P. Albee, "An Approach to Analyzing the Windows and Linux Security Models," in *Computer and Information Science, 2006 and 2006 1st IEEE/ACIS International Workshop on Component-Based Software Engineering, Software Architecture and Reuse. ICIS-COMSAR 2006. 5th IEEE/ACIS International Conference on*, 2006, pp. 56–62.

[29] L. Wang, T. Islam, T. Long, A. Singhal, and S. Jajodia, "An Attack Graph-Based Probabilistic Security Metric," in *Proceeedings of the 22Nd Annual IFIP WG 11.3 Working Conference on Data and Applications Security*, Berlin, Heidelberg, 2008, pp. 283–296.

[30] P. K. Manadhata and J. M. Wing, "An Attack Surface Metric," *IEEE Trans. Softw. Eng.*, vol. 37, no. 3, pp. 371–386, 2011.

[31] A. Agrawal, S. Chandra, and R. A. Khan, "An efficient measurement of object oriented design vulnerability," in *Availability, Reliability and Security, 2009. ARES'09. International Conference on*, 2009, pp. 618–623.

[32] Y. Shin and L. Williams, "An Empirical Model to Predict Security Vulnerabilities Using Code Complexity Metrics," in *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, New York, NY, USA, 2008, pp. 315–317.

[33] D. Subramanian, H. T. Le, P. K. K. Loh, and A. Premkumar, "An empirical vulnerability remediation model," in *Wireless Communications, Networking and Information Security (WCNIS), 2010 IEEE International Conference on*, 2010, pp. 376–380.

[34] R. Wang, L. Gao, Q. Sun, and D. Sun, "An Improved CVSS-based Vulnerability Scoring Mechanism," in *Proceedings of the 2011 Third International Conference on Multimedia Information Networking and Security*, Washington, DC, USA, 2011, pp. 352–355.

[35] J. L. Wright, M. McQueen, and L. Wellman, "Analyses of Two End-User Software Vulnerability Exposure Metrics," in *Proceedings of the 2012 Seventh International Conference on Availability, Reliability and Security*, Washington, DC, USA, 2012, pp. 1–10.

[36] V. S. Sharma and K. S. Trivedi, "Architecture Based Analysis of Performance, Reliability and Security of Software Systems," in *Proceedings of the 5th International Workshop on Software and Performance*, New York, NY, USA, 2005, pp. 217–227.

[37] O. Alhazmi, "Assessing Vulnerabilities in Software Systems: A Quantitative Approach," Colorado State University, Fort Collins, CO, USA, 2007.

[38] M. Almorsy, J. Grundy, and A. S. Ibrahim, "Automated Software Architecture Security Risk Analysis Using Formalized Signatures," in *Proceedings of the 2013 International Conference on Software Engineering*, Piscataway, NJ, USA, 2013, pp. 662–671.

[39] K. Sultan, A. En-Nouaary, and A. Hamou-Lhadj, "Catalog of Metrics for Assessing Security Risks of Software throughout the Software Development Life Cycle," in *Information Security and Assurance, 2008. ISA 2008. International Conference on*, 2008, pp. 461–465.

[40] P. Mell, K. Scarfone, and S. Romanosky, "Common Vulnerability Scoring System," *Secur. Priv. IEEE*, vol. 4, no. 6, pp. 85–89, Nov. 2006.

[41] Y. Shin, A. Meneely, L. Williams, and J. . Osborne, "Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities," *Softw. Eng. IEEE Trans. On*, vol. 37, no. 6, pp. 772–787, Nov. 2011.

[42] F. T. Sheldon, R. K. Abercrombie, and A. Mili, "Evaluating Security Controls Based on Key Performance Indicators and Stakeholder Mission," in *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, New York, NY, USA, 2008, pp. 41:1–41:11.

[43] R. Ortalo, Y. Deswarte, and M. Kaaniche, "Experimenting with quantitative evaluation tools for monitoring operational security," *Softw. Eng. IEEE Trans. On*, vol. 25, no. 5, pp. 633–650, Sep. 1999.

[44] K. Bernsmed and I. Tøndel, "Forewarned is Forearmed: Indicators for Evaluating Information Security Incident Management," in *IT Security Incident Management and IT Forensics (IMF), 2013 Seventh International Conference on*, 2013, pp. 3–14.

[45] L. Krautsevich, F. Martinelli, and A. Yautsiukhin, "Formal Analysis of Security Metrics with Defensive Actions," in *Ubiquitous Intelligence and Computing, 2013 IEEE 10th International Conference on and 10th International Conference on Autonomic and Trusted Computing (UIC/ATC)*, 2013, pp. 458–465.

[46] J. B. Michael, M.-T. Shing, K. J. Cruickshank, and P. J. Redmond, "Hazard Analysis and Validation Metrics Framework for System of Systems Software Safety," *Syst. J. IEEE*, vol. 4, no. 2, pp. 186–197, Jun. 2010.

[47] S. Ghaith and M. Ó Cinnéide, "Improving Software Security Using Search-based Refactoring," in *Proceedings of the 4th International Conference on Search Based Software Engineering*, Berlin, Heidelberg, 2012, pp. 121–135.

[48] Y. Shin, "Investigating Complexity Metrics As Indicators of Software Vulnerability," North Carolina State University, 2011.

[49] A. Meneely, "Investigating the Relationship Between Developer Collaboration and Software Security," North Carolina State University, 2011.

[50] Y. Shin and L. Williams, "Is Complexity Really the Enemy of Software Security?," in *Proceedings of the 4th ACM Workshop on Quality of Protection*, New York, NY, USA, 2008, pp. 47–50.

[51] L. Wang, S. Jajodia, A. Singhal, P. Cheng, and S. Noel, "k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities," *IEEE Trans. Dependable Secure Comput.*, vol. 11, no. 1, pp. 30–44, 2014.

[52] H. agen Hasle, Y. Kristiansen, K. Kintel, and E. Snekkenes, "Measuring Resistance to Social Engineering," in *Proceedings of the First International Conference on Information Security Practice and Experience*, Berlin, Heidelberg, 2005, pp. 132–143.

[53] P. Manadhata, J. Wing, M. Flynn, and M. McQueen, "Measuring the Attack Surfaces of Two FTP Daemons," in *Proceedings of the 2Nd ACM Workshop on Quality of Protection*, New York, NY, USA, 2006, pp. 3–10.

[54] K. Buyens, R. Scandariato, and W. Joosen, "Measuring the interplay of security principles in software architectures," *2013 ACM IEEE Int. Symp. Empir. Softw. Eng. Meas.*, vol. 0, pp. 554–563, 2009.

[55] W. H. Baker, L. P. Rees, and P. S. Tippett, "Necessary Measures: Metric-driven Information Security Risk Assessment and Decision Making," *Commun ACM*, vol. 50, no. 10, pp. 101–106, Oct. 2007.

[56] P. Anbalagan and M. Vouk, "On Reliability Analysis of Open Source Software - FEDORA," in *Software Reliability Engineering, 2008. ISSRE 2008. 19th International Symposium on*, 2008, pp. 325–326.

[57] L. Gallon, "On the Impact of Environmental Metrics on CVSS Scores," *Soc. Comput. IEEE Int. Conf. Priv. Secur. Risk Trust 2010 IEEE Int. Conf. On*, vol. 0, pp. 987–992, 2010.

[58] M. C. Gegick, "Predicting Attack-prone Components with Source Code Static Analyzers," North Carolina State University, 2009.

[59] V. H. Nguyen and L. M. S. Tran, "Predicting Vulnerable Software Components with Dependency Graphs," in *Proceedings of the 6th International Workshop on Security Measurements and Metrics*, New York, NY, USA, 2010, pp. 3:1–3:8.

[60] M. Gegick, L. Williams, J. Osborne, and M. Vouk, "Prioritizing Software Security Fortification Throughcode-level Metrics," in *Proceedings of the 4th ACM Workshop on Quality of Protection*, New York, NY, USA, 2008, pp. 31–38.

[61] M. Y. Liu, "Quantitative Security Analysis for Service-oriented Software Architectures," University of Victoria, Victoria, B.C., Canada, Canada, 2008.

[62] M. Li, J. Li, H. Song, and D. Wu, "Risk Management in the Trustworthy Software Process: A Novel Risk and Trustworthiness Measurement Model Framework," in *INC, IMS and IDC, 2009. NCM '09. Fifth International Joint Conference on*, 2009, pp. 214–219.

[63] J. Walden and M. Doyle, "SAVI: Static-Analysis Vulnerability Indicator," *IEEE Secur. Priv.*, vol. 10, no. 3, pp. 32–39, May 2012.

[64] T. Zimmermann, N. Nagappan, and L. Williams, "Searching for a Needle in a Haystack: Predicting Security Vulnerabilities for Windows Vista," in *Proceedings of the 2010 Third International Conference on Software Testing, Verification and Validation*, Washington, DC, USA, 2010, pp. 421–428.

[65] P. Parrend and S. Frenot, "Security benchmarks of OSGi platforms: toward Hardened OSGi," *Softw. Pract. Exp.*, vol. 39, no. 5, pp. 471–499, 2009.

[66] J. A. Wang, H. Wang, M. Guo, and M. Xia, "Security Metrics for Software Systems," in *Proceedings of the 47th Annual Southeast Regional Conference*, New York, NY, USA, 2009, pp. 47:1–47:6.

[67] I. Chowdhury, B. Chan, and M. Zulkernine, "Security Metrics for Source Code Structures," in *Proceedings of the Fourth International Workshop on Software Engineering for Secure Systems*, New York, NY, USA, 2008, pp. 57–64.

[68] A. Abdulrazeg, N. M. Norwawi, and N. Basir, "Security metrics to improve misuse case model," in *Cyber Security, Cyber Warfare and Digital Forensic (CyberSec), 2012 International Conference on*, 2012, pp. 94–99.

[69] J. Walden, M. Doyle, G. . Welch, and M. Whelan, "Security of open source web applications," in *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, 2009, pp. 545–553.

[70] J. Demme, R. Martin, A. Waksman, and S. Sethumadhavan, "Side-channel Vulnerability Factor: A Metric for Measuring Information Leakage," *SIGARCH Comput Arch. News*, vol. 40, no. 3, pp. 106–117, Jun. 2012.

[71] T. R. Gopalakrishnan Nair, V. Suma, and P. K. Tiwari, "Significance of depth of inspection and inspection performance metrics for consistent defect management in software industry," *Softw. IET*, vol. 6, no. 6, pp. 524–535, Dec. 2012.

[72] J. A. Wang, F. Zhang, and M. Xia, "Temporal Metrics for Software Vulnerabilities," in *Proceedings of the 4th Annual Workshop on Cyber Security and Information Intelligence Research: Developing Strategies to Meet the Cyber Security and Information Intelligence Challenges Ahead*, New York, NY, USA, 2008, pp. 44:1–44:3.

[73] H., Ph.D. Cavusoglu, B., Ph.D. Mishra, and S., Ph.D. Raghunathan, "The Effect of Internet Security Breach Announcements on Market Value: Capital Market Reactions for Breached Firms and Internet Security Developers," *Int J Electron Commer.*, vol. 9, no. 1, pp. 70–104, Oct. 2004.

[74] M. . Davidson, "The Good, the Bad, And the Ugly: Stepping on the Security Scale," in *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, 2009, pp. 187–195.

[75] M. Gegick, P. Rotella, and L. Williams, "Toward Non-security Failures As a Predictor of Security Faults and Failures," in *Proceedings of the 1st International Symposium on Engineering Secure Software and Systems*, Berlin, Heidelberg, 2009, pp. 135–149.

[76] A. Younis, Y. K. Malaiya, and I. Ray, "Using Attack Surface Entry Points and Reachability Analysis to Assess the Risk of Software Vulnerability Exploitability," in *High-Assurance Systems Engineering (HASE), 2014 IEEE 15th International Symposium on*, 2014, pp. 1–8.

[77] I. Chowdhury and M. Zulkernine, "Using Complexity, Coupling, and Cohesion Metrics As Early Indicators of

Vulnerabilities," *J Syst Arch.*, vol. 57, no. 3, pp. 294–313, Mar. 2011.

[78] Y. Beres, M. C. Mont, J. Griffin, and S. Shiu, "Using security metrics coupled with predictive modeling and simulation to assess security processes," in *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, 2009, pp. 564–573.

[79] A. Meneely, H. Srinivasan, A. Musa, A. Rodriguez Tejeda, M. Mokary, and B. Spates, "When a Patch Goes Bad: Exploring the Properties of Vulnerability-Contributing Commits," in *Empirical Software Engineering and Measurement, 2013 ACM / IEEE International Symposium on*, 2013, pp. 65–74.

APPENDIX A – METRIC NAMES AND PAPERS BY CATEGORY

Category	Definition
Churn (11)	30-Day Churn, Churn, Freq., LinesChanged, LinesNew, Lines deleted between revisions, NumCommits, % Interactive Churn, Relative Churn, Repeat Freq., Tot. Churn [P25, P33, P42, P44, P48, P53, P63]
Complexity (22)	Avg. Svc. Depth, Code Complex., Complex., CountPath complex., Cyclomatic Number, Dependency Network Complex., Exec. Complex., Fan In, Fan Out, Henry Kafura: SLOC * (FI*FO)^2), Lack of Cohesion of Meth.s, MaxFanIn, MaxFanOut, MaxMaxNesting, MaxNesting, Nesting complex., # Children, SumCyclomaticStrict, SumEssential, SumFanIn, SumFanOut, SumMaxNesting, V-Density - McCabe density, Weighted Meth.s per Class [P15, P25, P32, P34, P43, P45, P48, P53, P61]
Cost (5)	Annual Loss Expectancy, Cost of Sec. breach, Remediation Impact, Risk Potential, Threat-to-impact transitions [P11, P16, P39, P45, P57]
Coverage (100)	Access Accuracy, Administrator & op. logs, AnomalousSessionCount, Approval Accuracy, Arc cov., Audit logging, Block cov., Classif. Accessor Attr. Interactions, Classif. Attrs. Interaction Weight, Classif. Class Data Accessibility, Classif. Instance Data Accessibility, Classif. Meth. Extensibility, Classif. Meth.s Weight, Classif. Mutator Attr. Interactions, Classif. Op. Accessibility, Classif. Writing Meth.s Proportion, Composite-Part Crit. Classes, Ctrls. against malicious code, Countermeasure-effectiveness, Cov., Cov. of Hazard Analysis, Crit. Class Extensibility, Crit. Dsgn. Proportion, Crit. Element Ratio, Crit. Serialized Classes Proportion, Crit. Superclasses Inher., Crit. Superclasses Proportion, Depth of inspect., eir - ratio between extern. & intern. data flow, Fail-Safe Defaults, Grant Least Priv., Hazard Analysis Achieved, Isolation, Least Cmn. Mech., # Catch Blocks Per Class, % High-Risk Softw. Hazards with Safety Rqts., % Medium-Risk Softw. Hazards with Safety Rqts., % Moderate-Risk Softw. Hazards with Safety Rqts., % Softw. Safety Rqts., % Softw. Safety Rqts. Traceable to Hazards, % identified corrective action not impl., % incidents a recurrence of previous incidents, % increases in reported Sec. incidents,% IT assets for which fault tol. mechs. have been impl., % IT assets for which recovery procedures have been defined & impl., % IT assets for which redundancy mechs. have been impl., % new comps. deployed in the sys. all at once, % Org. attended Sec. training, % Org. contributing to dev., % reported incidents followed up & mitigated, % reported Sec. incidents where the cause of the incident was identified, % Sec. incidents related to incorrect, incomplete, missing or compromised audit data, % Sec. incidents related to lack of an audit capability, % Sec. incidents related to the ability to bypass an audit function, % Sec. incidents that exploited existing Vuln. with known soln.s, % servers with installed & active auto. hard-disk encryption, % sys. arch. for which failure modes have been thoroughly identified, % sys. changes reviewed for Sec. impacts before impl., % sys. for which appr. config. settings has been impl., % sys. that depends on extern. comps. that the Org. lacks ctrl. over, % sys. subject to risk analysis, % sys. continuously monitored for config. policy compliance, % sys. where permissions to install non-standard softw. is ltd. or prohibited, % sys. where the auth. to make config. changes is ltd. in accordance to policy, % systems with the latest appr. patches installed, % systems exposed at time of malware, PercentValidatedInput, Ratio Dsgn. Decisions, Ratio Dsgn. Flaws Related to Sec., ratio extension misuse cases extended once to the tot. # extension misuse cases, Ratio Impl. Err.s That Have an Impact on Sec., ratio inclusion misuse cases incl. once to the tot. # inclusion misuse cases, ratio misuse cases used as pre/post conditions of other misuse cases to the tot. # misuse cases, Ratio Patches Issued to Address Sec. Vuln., Ratio Sec. Rqts., Ratio Sec. Test Cases, Ratio Sec. Test Cases that Fail, Ratio Shared Resources, Ratio Softw. Changes Due to Sec. Considerations, ratio the # incl. Sec. Rqts. to the tot. # stated Sec. Rqts., ratio the # misuse cases that do not threaten the application to the tot. # misuse cases, Ratio the # Omitted Exceptions, Ratio the # Omitted Sec. Rqts., ratio the # the base misuse cases associated to one misuser to the tot. # base misuse cases, ratio the # unmitigated misuse cases that threaten the application to the tot. # misuse cases, Readability of Classif. Attrs., Readability via Classif. Meth.s, Readability via Crit. Classes, Secure the Weakest Link, Softw. Hazard Analysis Depth, Static analysis alert density, static analysis Vuln. density, Target Distribution, Unaccessed Assigned Classif. Attr., Uncalled Classif. Accessor Meth., Unused Crit. Accessor Class, Window of exposure, Writability of Classif. Attrs., Writability via Classif. Meth.s, Writability via Crit. Classes [P1, P17, P19, P2, P23, P24, P28, P30, P31, P39, P4, P41, P43, P44, P45, P47, P48, P51, P52, P53, P55, P58, P60, P62]
Dependency (33)	Avg. intern. data flow, Coupling, Coupling between comps., Coupling Between Object classes, Coupling Between Objects, Coupling Corruption Propagation, Crit. Classes Coupling, edf - Avg. extern. data flow, edfin - Avg. inc. data flow, edfin - inc. data flow of, edfout - Avg. out. data flow, edfout - out. data flow of, Eigenvector Centrality, Flow_ Betweenness, ie - # edges between two arbitrary nodes, Inc. closure, Inc. direct, InDegree, InDegree_w, Intern. data flow, Lack of Cohesion of Meth.s, Layer info, NumCalls, oir - ratio between Avg. out. & inc. data flow, Op. Env. Sec. Measurement, OutDegree, OutDegree_w, Out.

	closure, Out. direct, Paths, Prog. Impl. Sec. Measurement, Reflection Package Boolean, RW_ Betweenness, Sec. Indictor of Softw. Sys., Sec. Metrics of Arith. Expr., Sec. Metrics of Array Index, Sec. Metrics of Comp. Interf., Sec. Metrics of Ctrl. Op., Sec. Metrics of I/O Mgmt, Sec. Metrics of Input Fmt., Sec. Metrics of Kernel Op., Sec. Metrics of Network Env., Sec. Metrics of Resource Alloc., Sec. Metrics of User Auth., SP_Betweenness, Vuln. Propagation Factor, Classif. Attrs. Inher., Classif. Meth.s Inher., Depth of Inher. Tree [P1, P14, P31, P32, P34, P42, P43, P45, P48, P51, P61]
Effort (23)	Access Complex., Access Vector, Adversary Work Factor, Attackability, Authentication, Conf. Rqt., Dmg. potential-effort ratio, Depth, ExclusiveExeTime, Expected Time to Completion, Exploitability, InclusiveExeTime, Integrity Rqt., Mean Effort to Sec. Failure, Min. cost of attack, Min. length of attack, Prot. Rate, Rigor, Side-chan. Vuln. Factor, Social Eng. Resistance, Struc. sev., Vuln. exploitability, Weakest adversary [P17, P19, P20, P24, P27, P29, P3, P32, P36, P41, P49, P54, P56, P6, P60, P8, P9]
Organization (18)	CNBetweenness, CNCloseness, Depth of Master Ownership, DNAvgBetweenness, DNAvgCloseness, DNAvgEdgeBetweenness, DNMaxCloseness, DNMaxDegree, DNMaxEdgeBetweenness, DNMinBetweenness, DNMinDegree, Edit Freq., Lvl. of Org. Code Ownership, New Effective Author, # Ex-Engineers, # Developers, Org. Intersection Factor, Overall Org. Ownership [P25, P33, P48, P63]
Size (50)	LOC, Instr. count, Arith. Expr., array index, Attack Surface Metric, blank lines, Classif. Attrs. Tot., Classif. Meths. Tot., Comment Lines, comp. interf., ctrl. op., # Base Classes, Crit. Classes Tot., data fmt., Economy of Mech., I/O mgmt, Interf. complex. density, kernel ctrl., LOC, LOCVarDecl, network planning, # Attacks, # data items transferred in an edge, # Dsgn. Decisions Related to Sec., # Developers, # member nodes, # params. in the meth. sig., # published Vuln., # return points in the meth., # Sec. Algs., # Sec. Incidents Reported, # sub classes, NumFunctions, NumLineProcessor, Paths, Reduce Attack Surface, resource alloc., Response for a Class, rin - # inc. cxns., rout - # out. cxns. from, Sec. Abs. Measurements, Stall Ratio, Tot. global vars, tot. # elicited Sec. use cases, tot. # excl. Sec. Rqts. that ensure session handling, tot. # excl. Sec. Rqts. that put the sys. at risk of possible attacks, tot. # identified misuse cases, Tot. # Sec. Rqts., Tot. Sec. Index, user auth., Vol. of email correspondence with Vuln. handling team, Weighted Meth.s per Class [P1, P10, P13, P15, P23, P25, P29, P32, P34, P37, P38, P42, P43, P44, P48, P51, P52, P53, P58, P61, P7]
Strength (30)	Availability Impact, Availability Rqt., CMMI Lvl., Comment ratio, CommentDensity, Compartmentalization, Completeness of fix, Conf., Conf. Impact, Defense-In-Depth, Expected Reliability, Fail Securely, Indep. of Verification, Inspect. perf., Integrity, Isolation, k-zero day safety, Least Priv., Mean Time To Failure, Rpt. Confidence, risk ctrl., risk identification, Sec. resource indicator, Svc. Mech. Str., softw. proj. mgmt exp., Trustworthiness, Use of (automated) tools, Variable Sec. Vuln., Vuln. Confidence Coefficient, Vuln. Free Days [P16, P17, P18, P19, P20, P22, P24, P25, P32, P35, P41, P45, P46, P5, P53, P55, P56, P58, P60, P61, P9]
Weakness (55)	Attack Graph Prob., Avg. Active Vuln. per day, Avg. time from incident det. until incident has been reported, BrokenAccountCount, Collateral Dmg. Potential, CVSS Base Score, Developer Risk, Dev. Risk, Env. Risk, Excess Priv., Expected threat Freq., Expected Vuln., ExploitedFlawCount, Faults found during manual inspect., InjectionFlawCount, Integrity Impact, Max. prob. of successful attack, Mean Failure Cost, Mean time from vendor patch availability to patch approval to patch installation, Mean time to incident det., Monitoring sys. use, non-Sec. failure reports, # Dsgn. Flaws Related to Sec., # Exceptions Impl. to Handle Exec. Failures Related to Sec., # excl. Sec. Rqts. that ensure i/o handling, # function calls that don't check return values, # Impl. Err.s Found in the Sys., # Impl. Err.s Related to Sec., # Omitted Exceptions for Handling Exec. Failures Related to Sec., # Omitted Sec. Rqts., # open Sec. bugs, # reported Sec. incidents, # Sec. bulletins issues per yr., # Svc. accts. with weak or default passwords, # successful attempts to execute recovery this period, # violations of the LP principle, OverflowVulnCount, % Softw. Hazards, Proj. Mgmt Risk, Remediation Lvl., Remediation Potency, Remediation Scheme, Rqts. Risk, Sec. of sys. doc., Static analysis alert count, Temporal Score, Time to close bug/Vuln., Time to Problem Correction, Time to Problem Rpt., User Risk, Vuln. found during Rqts., dsgn. & coding, Vuln. found post-dev., Vuln. Density, Weakness, XsiteVulnCount, "The Sec. Metric", Attack-Reward (URL Jumping), Avg. Svc. Depth, Classif. Accessor Attr. Interactions, Classif. Attrs. Inher., Classif. Attrs. Interaction Weight, Classif. Class Data Accessibility, Classif. Instance Data Accessibility, Classif. Meth. Extensibility, Classif. Meth.s Extensibiliity, Classif. Meth.s Inher., Classif. Meth.s Weight, Classif. Mutator Attr. Interactions, Classif. Op. Accessibility, CNBetweenness, Composite-Part Crit. Classes, Conf., Crit. Class Extensibility, Crit. Dsgn. Proportion, Crit. Superclasses Inher., Crit. Superclasses Proportion, Dsgn. Size, DNMaxEdgeBetweenness, Knot Count, Kolmogorov Complex., Measurement of Cost, NumCommits, NumDevs, Potency, Resilience, Shortest Path [P12, P16, P17, P18, P2, P20, P21, P23, P24, P26, P28, P29, P30, P38, P39, P4, P40, P41, P42, P44, P45, P46, P50, P52, P56, P58, P59, P60, P9]

Appendix B: List of Selected Papers

Paper #	Paper Title
P1[19]	A Hierarchical Security Assessment Model for Object-Oriented Programs
P2[20]	A Metrics Framework to Drive Application Security Improvement
P3[11]	A Near Real-time System for Security Assurance Assessment
P4[21]	A New Method for the Identification of Proactive Information Security Management System Metrics
P5[22]	A New Security Sensitivity Measurement for Software Variables
P6[23]	A Weakest-Adversary Security Metric for Network Configuration Security Analysis
P7[24]	Adaptive and quantitative comparison of J2EE vs. .NET based on attack surface metric
P8[25]	Adversary Work Factor as a Metric for Information Assurance
P9[26]	An analysis of CVSS version 2 vulnerability scoring
P10[27]	An Analyzer-Based Security Measurement Model for Increasing Software Security
P11[28]	An Approach to Analyzing the Windows and Linux Security Models
P12[29]	An Attack Graph-Based Probabilistic Security Metric
P13[30]	An Attack Surface Metric
P14[31]	An Efficient Measurement of Object Oriented Design Vulnerability
P15[32]	An empirical model to predict security vulnerabilities using code complexity metrics
P16[33]	An empirical vulnerability remediation model
P17[34]	An Improved CVSS-based Vulnerability Scoring Mechanism
P18[35]	Analyses of Two End-User Software Vulnerability Exposure Metrics
P19[13]	Appraisal and reporting of security assurance at operational systems level
P20[36]	Architecture based analysis of performance, reliability and security of software systems
P21[37]	Assessing vulnerabilities in software systems: a quantitative approach
P22[38]	Automated Software Architecture Security Risk Analysis using Formalized Signatures
P23[39]	Catalog of Metrics for Assessing Security Risks of Software throughout the Software Development Life Cycle
P24[40]	Common Vulnerability Scoring System
P25[41]	Evaluating Complexity, Code Churn, and Developer Activity Metrics as Indicators of Software Vulnerabilities
P26[42]	Evaluating Security Controls Based on Key Performance Indicators and Stakeholder Mission
P27[43]	Experimenting with Quantitative Evaluation Tools for Monitoring Operational Security
P28[44]	Forewarned is Forearmed: Indicators for Evaluating Information Security Incident Management
P29[45]	Formal Analysis of Security Metrics with Defensive Actions
P30[46]	Hazard Analysis and Validation Metrics Framework for System of Systems Software Safety
P31[47]	Improving Software Security Using Search-Based Refactoring
P32[48]	Investigating Complexity Metrics as Indicators of Software Vulnerability
P33[49]	Investigating the Relationship Between Developer Collaboration and Software Security
P34[50]	Is Complexity Really the Enemy of Software Security?
P35[51]	k-Zero Day Safety: A Network Security Metric for Measuring the Risk of Unknown Vulnerabilities
P36[52]	Measuring Resistance to Social Engineering
P37[53]	Measuring the Attack Surfaces of Two FTP Daemons
P38[54]	Measuring the Interplay of Security Principles in Software Architectures
P39[55]	Necessary Measures: Metric-Driven Information Security Risk Assessment and Decision Making

P40[56]	On Reliability Analysis of Open Source Software - FEDORA
P41[57]	On the Impact of Environmental Metrics on CVSS Scores
P42[58]	Predicting Attack-Prone Components with Source Code Static Analyzers
P43[59]	Predicting Vulnerable Software Components with Dependency Graphs
P44[60]	Prioritizing Software Security Fortification Through Code-Level Metrics
P45[61]	Quantitative Security Analysis for Service-Oriented Software Architectures
P46[62]	Risk Management in the Trustworthy Software Process: A Novel Risk and Trustworthiness Measurement Model Framework
P47[63]	SAVI: Static-Analysis Vulnerability Indicator
P48[64]	Searching for a Needle in a Haystack: Predicting Security Vulnerabilities for Windows Vista
P49[65]	Security Benchmarks of OSGi Platforms: Toward Hardened OSGi.
P50[66]	Security Metrics for Software Systems
P51[67]	Security Metrics for Source Code Structures
P52[68]	Security Metrics to Improve Misuse Case Model
P53[69]	Security of Open SourceWeb Applications
P54[70]	Side-Channel Vulnerability Factor: A Metric for Measuring Information Leakage
P55[71]	Significance of Depth of Inspection and Inspection Performance Metrics for Consistent Defect Management in Software Industry
P56[72]	Temporal Metrics for Software Vulnerabilities
P57[73]	The Effect of Internet Security Breach Announcements on Market Value
P58[74]	The Good, The Bad, And The Ugly: Stepping on the Security Scale
P59[75]	Toward Non-security Failures as a Predictor of Security Faults and Failures
P60[76]	Using Attack Surface Entry Points and Reachability Analysis to Assess the Risk of Software Vulnerability Exploitability
P61[77]	Using Complexity, Coupling, and Cohesion Metrics as Early Indicators of Vulnerabilities
P62[78]	Using Security Metrics Coupled with Predictive Modeling and Simulation to Assess Security Processes
P63[79]	When a Patch Goes Bad: Exploring the Properties of Vulnerability-Contributing Commits

Appendix C: Metrics Identified

M1	30-Day Churn
M2	Access Accuracy
M3	Access Complexity (AC)
M4	Access Vector (AV)
M5	Administrator and operator logs
M6	Adversary Work Factor
M7	Annual Loss Expectancy (ALE)
M8	AnomalousSessionCount
M9	Approval Accuracy
M10	Arc coverage
M11	Arithmetic Expression
M12	Array index
M13	Attack Graph Probability
M14	Attack Surface Metric
M15	Attackability
M16	Audit logging
M17	Authentication (AU)
M18	Availability Impact (A)
M19	Availability Requirement (AR)
M20	Average Active Vulnerabilities per day (AAV)
M21	Average internal data flow (AIDF)
M22	Average Service Depth (ASD)
M23	Average time from incident detection until incident has been reported -
M24	Blank lines
M25	Block coverage
M26	BrokenAccountCount
M27	Churn
M28	Classified Accessor Attribute Interactions (CAAI)
M29	Classified Attributes Inheritance (CAI)
M30	Classified Attributes Interaction Weight (CAIW)
M31	Classified Attributes Total (CAT)
M32	Classified Class Data Accessibility (CCDA)
M33	Classified Instance Data Accessibility (CIDA)
M34	Classified Method Extensibility (CME)
M35	Classified Methods Inheritance (CMI)
M36	Classified Methods Total (CMT)
M37	Classified Methods Weight (CMW)
M38	Classified Mutator Attribute Interactions (CMAI)
M39	Classified Operation Accessibility (COA)
M40	Classified Writing Methods Proportion (CWMP)
M41	CMMI Level

M42	CNBetweenness
M43	CNCloseness
M44	Code Complexity
M45	Collateral Damage Potential (CDP)
M46	Comment Lines
M47	Comment ratio
M48	CommentDensity
M49	Compartmentalization
M50	Completeness of fix
M51	Complexity
M52	Component interface
M53	Composite-Part Critical Classes (CPCC)
M54	Confidentiality
M55	Confidentiality Impact (C)
M56	Confidentiality Requirement (CR)
M57	Control operation
M58	Controls against malicious code
M59	Cost of security breach
M60	Count of Base Classes (CBC)
M61	Countermeasure-effectiveness
M62	CountPath complexity
M63	Coupling
M64	Coupling between components
M65	Coupling Between Object classes (CBOC)
M66	Coupling Between Objects (CBO)
M67	Coupling Corruption Propagation
M68	Coverage
M69	Coverage of Hazard Analysis
M70	Critical Class Extensibility (CCE)
M71	Critical Classes Coupling (CCC)
M72	Critical Classes Total (CCT)
M73	Critical Design Proportion (CDP)
M74	Critical Element Ratio
M75	Critical Serialized Classes Proportion (CSCP)
M76	Critical Superclasses Inheritance (CSI)
M77	Critical Superclasses Proportion (CSP)
M78	CVSS Base Score
M79	Damage potential-effort ratio
M80	Data format
M81	Defense-In-Depth
M82	Dependency Network Complexity
M83	Depth

M84	Depth of Inheritance Tree (DIT)
M85	Depth of inspection
M86	Depth of Master Ownership
M87	Developer Risk
M88	Development Risk
M89	DNAvgBetweenness
M90	DNAvgCloseness
M91	DNAvgEdgeBetweenness
M92	DNMaxCloseness
M93	DNMaxDegree
M94	DNMaxEdgeBetweenness
M95	DNMinBetweenness
M96	DNMinDegree
M97	Economy of Mechanism (PEM)
M98	EDF - average external data flow
M99	EDFIN - average incoming data flow
M100	EDFIN - incoming data flow of
M101	EDFOUT - average outgoing data flow
M102	EDFOUT - outgoing data flow of
M103	Edit Frequency
M104	Eigenvector Centrality (EvCent)
M105	EIR - ratio between external and internal data flow
M106	Environment Risk
M107	Excess Privilege
M108	ExclusiveExeTime
M109	Execution Complexity
M110	Expected Reliability
M111	Expected threat frequency
M112	Expected Time to Completion
M113	Expected Vulnerability
M114	Exploitability (TE)
M115	ExploitedFlawCount
M116	Fail Securely
M117	Fail-Safe Defaults (PFSD)
M118	Fan In (FI)
M119	Fan Out (FO)
M120	Faults found during manual inspection
M121	Flow_ Betweenness
M122	Frequency
M123	Grant Least Privilege (PLP)
M124	Hazard Analysis Achieved
M125	HK (Henry Kafura: SLOC * (FI*FO)^2)

M126	I/O management
M127	IE - number of edges between two arbitrary nodes
M128	InclusiveExeTime
M129	Incoming closure
M130	Incoming direct
M131	InDegree
M132	InDegree_w
M133	Independence of Verification
M134	InjectionFlawCount
M135	Inspection performance
M136	Integrity
M137	Integrity Impact (I)
M138	Integrity Requirement (IR)
M139	Interface complexity density (I-density)
M140	Internal data flow (IDF)
M141	Isolation (PI)
M142	K-zero day safety
M143	Kernel control
M144	Lack of Cohesion of Methods (LCOM)
M145	Layer information
M146	Least Common Mechanism (PLCM)
M147	Least Privilege
M148	Level of Organizational Code Ownership
M149	Lines of Code (LOC)
M150	LinesChanged
M151	LinesNew
M152	LOCVarDecl
M153	MaxFanIn
M154	MaxFanOut
M155	Maximal probability of successful attack
M156	MaxMaxNesting
M157	MaxNesting
M158	Mean Effort to security Failure
M159	Mean Failure Cost
M160	Mean time from vendor patch availability to patch approval to patch installation
M161	Mean Time To Failure
M162	Mean time to incident detection
M163	Minimal cost of attack
M164	Minimal length of attack
M165	Monitoring system use
M166	Nesting complexity
M167	Network planning

M168	New Effective Author (NEA)
M169	Non-security failure reports
M170	Number 0f Ex-Engineers
M171	Number of Attacks
M172	Number of Catch Blocks Per Class
M173	Number Of Children (NOC)
M174	Number of data items transferred in an edge
M175	Number of Design Decisions Related to Security (NDD)
M176	Number of Design Flaws Related to Security (NSDF)
M177	Number of Developers
M178	Number of Exceptions That Have Been Implemented to Handle Execution Failures Related to Security (NEX)
M179	Number of excluded security requirements that ensure input/output handling
M180	Number of function calls that don't check return values
M181	Number of Implementation Errors Found in the System (NERR)
M182	Number of Implementation Errors Related to Security (NSERR)
M183	Number of lines deleted between revisions
M184	Number of member nodes
M185	Number of Omitted Exceptions for Handling Execution Failures Related to Security (NOEX)
M186	Number of Omitted Security Requirements (NOSR)
M187	Number of open security bugs
M188	Number of parameters in the method signature
M189	Number of published vulnerabilities
M190	Number of reported security incidents
M191	Number of return points in the method
M192	Number of Security Algorithms (NSA)
M193	Number of security bulletins issues per year
M194	Number of Security Incidents Reported (NSR)
M195	Number of service accounts with weak or default passwords
M196	Number of sub classes
M197	Number of successful attempts to execute recovery this period
M198	Number of violations of the LP principle
M199	NumCalls
M200	NumCommits
M201	NumFunctions
M202	NumLineProcessor
M203	OIR - ratio between average outgoing and incoming data flow
M204	Organization Intersection Factor
M205	OutDegree
M206	OutDegree_w
M207	Outgoing closure
M208	Outgoing direct
M209	Overall Organization Ownership

M210	OverflowVulnCount
M211	Paths
M212	Percent High-Risk Software Hazards with Safety Requirements
M213	Percent Medium-Risk Software Hazards with Safety Requirements
M214	Percent Moderate-Risk Software Hazards with Safety Requirements
M215	Percent Software Hazards
M216	Percent Software Safety Requirements
M217	Percent Software Safety Requirements Traceable to Hazards
M218	Percentage Interactive Churn (PIC)
M219	Percentage of identified corrective action that has not been implemented
M220	Percentage of incidents that are a recurrence of previous incidents
M221	Percentage of increases in reported security incidents -
M222	Percentage of IT assets for which fault tolerance mechanisms have been implemented
M223	Percentage of IT assets for which recovery procedures have been defined and implemented
M224	Percentage of IT assets for which redundancy mechanisms have been implemented
M225	Percentage of new components that were deployed in the system all at once
M226	Percentage of organization attended security training
M227	Percentage of Organization contributing to development
M228	Percentage of reported incidents that have been followed up and mitigated
M229	Percentage of reported security incidents where the cause of the incident was identified
M230	Percentage of security incidents related to incorrect, incomplete, missing or compromised audit data
M231	Percentage of security incidents related to lack of an audit capability
M232	Percentage of security incidents related to the ability to bypass an audit function
M233	Percentage of security incidents that exploited existing vulnerabilities with known solutions
M234	Percentage of servers with installed and active automatic hard-disk encryption
M235	Percentage of system architecture for which failure modes have been thoroughly identified
M236	Percentage of system changes that were reviewed for security impacts before implementation
M237	Percentage of system for which approved configuration settings has been implemented
M238	Percentage of system that depends on external components that the organization lacks control over
M239	Percentage of system that has been subject to risk analysis -
M240	Percentage of system that is continuously monitored for configuration policy compliance
M241	Percentage of system where permissions to install non-standard software is limited or prohibited
M242	Percentage of system where the authority to make configuration changes is limited in accordance to policy
M243	Percentage of systems with the latest approved patches installed
M244	Percentage of sytems exposed at time of malware
M245	PercentValidatedInput
M246	Project Management Risk
M247	Protection Rate (PR)
M248	Ratio of Design Decisions (RDD)
M249	Ratio of Design Flaws Related to Security (RDF)
M250	Ratio of extension misuse cases extended once to the total number of extension misuse cases.
M251	Ratio of Implementation Errors That Have an Impact on Security (RSERR)

M252	Ratio of inclusion misuse cases included once to the total number of inclusion misuse cases.
M253	Ratio of misuse cases used as pre/post conditions of other misuse cases to the total number of misuse cases.
M254	Ratio of Patches Issued to Address Security Vulnerabilities (RP)
M255	Ratio of Security Requirements (RSR)
M256	Ratio of Security Test Cases (RTC)
M257	Ratio of Security Test Cases that Fail (RTCP) (sic)
M258	Ratio of Shared Resources (RSR)
M259	Ratio of Software Changes Due to Security Considerations (RSC)
M260	Ratio of the number of included security requirements to the total number of stated security requirements
M261	Ratio of the number of misuse cases that do not threaten the application to the total number of misuse cases.
M262	Ratio of the Number of Omitted Exceptions (ROEX)
M263	Ratio of the Number of Omitted Security Requirements (ROSR)
M264	Ratio of the number of the base misuse cases associated to one misuser to the total number of base misuse cases.
M265	Ratio of the number of unmitigated misuse cases that threaten the application to the total number of misuse cases.
M266	Readability of Classified Attributes (RCA)
M267	Readability via Classified Methods (RCM)
M268	Readability via Critical Classes (RCC)
M269	Reduce Attack Surface (PRAS)
M270	Reflection Package Boolean (RPB)
M271	Relative Churn
M272	Remediation Impact (RJ)
M273	Remediation Level (RL)
M274	Remediation Potency (RP)
M275	Remediation Scheme (RS)
M276	Repeat Frequency
M277	Report Confidence (RC)
M278	Requirements Risk
M279	Resource allocation
M280	Response for a Class (RFC)
M281	Rigor
M282	RIN - number of incoming connections
M283	Risk control
M284	Risk identification
M285	Risk Potential
M286	ROUT - number of outgoing connections from
M287	RW_ Betweenness
M288	Secure the Weakest Link (PSWL)
M289	Security Absolute Measurements (SAM)
M290	Security of system documentation
M291	Security resource indicator
M292	Service Mechanism Strength
M293	Side-channel Vulnerability Factor

M294	Social Engineering Resistance (SER)
M295	Software Hazard Analysis Depth
M296	Software project management experience
M297	SP_Betweenness
M298	Stall Ratio
M299	Static analysis alert count
M300	Static analysis alert density
M301	Static analysis vulnerability density
M302	Structural severity
M303	SumCyclomaticStrict
M304	SumEssential
M305	SumFanIn
M306	SumFanOut
M307	SumMaxNesting
M308	Target Distribution (TD)
M309	Temporal Score
M310	Threat-to-impact transitions
M311	Time to close bug/vulnerability
M312	Time to Problem Correction
M313	Time to Problem Report
M314	Total Churn
M315	Total global variables
M316	Total number of elicited security use cases
M317	Total number of excluded security requirements that ensure session handling
M318	Total number of excluded security requirements that put the system at risk of possible attacks.
M319	Total number of identified misuse cases
M320	Total Number of Security Requirements (NSR)
M321	Total Security Index (TSI)
M322	Trustworthiness
M323	Unaccessed Assigned Classified Attribute (UACA)
M324	Uncalled Classified Accessor Method (UCAM)
M325	Unused Critical Accessor Class (UCAC)
M326	Use of (automated) tools
M327	User authority
M328	User Risk
M329	V-Density - McCabe density
M330	Variable Security Vulnerability
M331	Volume of email correspondence with vulnerability handling team
M332	Vulnerabilities found during requirements, design and coding
M333	Vulnerabilities found post-development
M334	Vulnerability Confidence Coefficient (VCC)
M335	Vulnerability Density

M336	Vulnerability exploitability
M337	Vulnerability Free Days (VFD)
M338	Vulnerability Propagation Factor (VPF)
M339	Weakest adversary
M340	Weakness
M341	Weighted Methods per Class (WMC)
M342	Window of exposure
M343	Writability of Classified Attributes (WCA)
M344	Writability via Classified Methods (WCM)
M345	Writability via Critical Classes (WCC)
M346	XsiteVulnCount