American Journal of Software Engineering and Applications
2013; 2(6): 150-155
Published online December 20, 2013 (http://www.sciencepublishinggroup.com/j/ajsea)
doi: 10.11648/j.ajsea.20130206.14

Software security metric development framework (an early stage approach)

A. Agrawal[1, *], R. A. Khan[2]

[1]Department of Computer Science, Khwaja Moinuddin Chishti Urdu, Arabi-Farsi University, Lucknow, India
[2]Department of IT, Babasaheb Bhimrao Ambedkar University, Lucknow, India

Email address:
alka_csjmu@yahoo.co.in (A. Agrawal), khanraees@yahoo.com (R. A. Khan)

To cite this article:
A. Agrawal, R. A. Khan. Software Security Metric Development Framework (An Early Stage Approach). *American Journal of Software Engineering and Applications*. Vol. 2, No. 6, 2013, pp. 150-155. doi: 10.11648/j.ajsea.20130206.14

Abstract: This paper does an extensive survey on software security metrics and put forth an effort to characterize design time software security. Misconceptions associated to security metrics have been identified and discussed. A list of characteristics good security metrics should posses is listed. In absence of any standard guideline or methodology to develop early stage security metrics, an effort has been made to provide a strong theoretical basis to develop such a framework. As a result, a Security Metrics Development Framework has been proposed in this paper. Our next effort will be to implement the proposed framework to develop security metrics in early stage of software development life cycle.

Keywords: Software Security, Software Security Metrics, Metric Development, Design Phase

1. Introduction

The increasing use of information system led to dramatically improve the functionality with respect to safety, cost and reliability. The exponential growth of technology and the prospect of increased public access to the computing, communications, and storage resources have made these systems more vulnerable to attacks. A system cannot be considered as of high assurance if it has poor security. Security problems involving computers and software are frequent, widespread, and serious. In an era riddled with asymmetric cyber attacks, claims about system reliability, integrity and safety must also include provisions for built-in security of the enabling software.

Generally, software developed and implemented has bugs, and many of the bugs available with the software system have security implications. As reported by various researchers and practitioners, security incidents seem to have increased exponentially. Security engineering as a discipline is still in its infancy. The field is hampered by its lack of adequate measures of goodness. Without such a measure, it is difficult to judge progress and it is particularly difficult to make engineering trade-off decisions when designing systems [1]. A widely accepted management principle is that an activity cannot be managed if it cannot be measured. Software security also falls in this rubric [2]. All security vulnerabilities in software are the result of security bugs or defects within the software. In most cases, these defects are created by two primary causes including non-conformance, or a failure to satisfy requirements, and an error or omission in the software requirements.

Software security is a concept that still lacks unambiguous definitions. It is important to understand the nature of software when developing methods for measuring software security. A common way to try to understand software security is to find different dimensions of it including confidentiality, integrity and availability [3]. Most commercial software suffers from significant design and implementation security vulnerabilities because of two most important factors including complexity and motivation. Software developers are producing more complex software and work constantly on the boundary of manageable complexity. Most of the software contains security flaws because of the complex nature. Developers are readily capable of preventing them. The second cause of software insecurity is because of the lack of motivation to the vendors for creating more secure software as the economics of the software industry provide them with little incentive [4].

Security vulnerabilities are increasingly due to software. Researchers and practitioners have carried out much work on code-level vulnerabilities including buffer overflows. But, at the same time, there is a great demand in identifying and mitigating security vulnerabilities at design level [5]. In August 2006, first-time Steve Bellovin, argued that for software, meaningful security metrics are not yet possible because 100 percent security of software is not possible, i.e., one cannot measure what cannot possible exist [6][7]. Regulatory, financial, and organizational reasons drive the requirement to measure software security performance. Software security metrics provide a practical approach to measuring security by facilitating decision making and accountability through collection, analysis, and reporting of relevant performance data [8].

2. Security Measurement

Measurement is a decision aid and what needs to be measured depends on the decision. Measurement in any science and engineering can be done by involving three main steps including data collection, data validation, and data processing. Data collection defines what to collect and how to collect the data. The kind of data to be collected is directly linked to the kind of behavior to be analyzed and to the quantitative measures to be evaluated to characterize such behavior. Data validation analyzes the collected data for correctness, consistency, and completeness. Data processing performs statistical analysis on the validated data to identify and analyze trends and to evaluate quantitative measures that characterize security [9]. Software security measurement requires [10]:

- Identifying measurable security characteristics;
- Specify security metrics to be utilized;
- Map identified measurable characteristics to security metrics;
- Associate sub-sets of security characteristics to software system entities;
- Develop or use methodology to assess security strength of system entities.

There is a noticeable difference between metrics and measurements. Measurements provide single-point-in-time views of specific, discrete factors. On the other hand, metrics are derived from comparing two or more measurements taken over time with a predetermined baseline [11]. Alger differentiates measurements from metrics and believes that measurements are generated by counting, whereas metrics are generated from analysis [12]. Software measurement is at the foundation of software engineering. Software security metrics are quantitative measurements that are important for assessing the effects of proposed improvements in security engineering. Metrics serve an equally important role in risk analysis, scheduling, planning, resource allocation, and cost estimation. This results in implications on what should and may be measured.

Actual Measurable: Security metrics are fundamental in order to specify what is actually to be measured. In a simplified manner, a metric may be defined as a framework in which raw data (measurements) are given a signification or meaning.

Aggregation: There is a common agreement between researchers and practitioners that there is no single measure available to capture the security value of software system. Thus, security measurement methods have to be able to combine several measurements into software system wide-values.

A security metric measures or assesses the extent to which a system meets its security objectives. Since meaningful quantitative security metrics are largely unavailable, the security community primarily uses qualitative metrics for security.

3. Security Metrics

Building secure software highly depends on quantitative measurement of software security. Security measurement defines the target security level and achievable security levels. Metrics and measurements are the cornerstones of any scientific discipline [13]. Software security measurement is essential in order to make good decisions about how to design security countermeasures. A measure is a dimension compared against a standard. Security measures assists in choosing alternative security architectures, and improving security during design and operations [14].

It is essential to be able to define the actual meaning while security is measured. Security metrics is a term that has been used for the purpose. The need and significance of security metrics has been emphasized by researchers [15] [16] as well as by the industry practitioners [17]. A metric is a system of related measures enabling quantification of some characteristic. Security metrics are essential to meeting organizations security objectives. Without good security metrics, it is very difficult to assert a certain level of security [13]. A security metric is a system of related dimensions enabling quantification of the degree of freedom from possibility of suffering damage or loss from malicious attack [14]. An exhaustive review of literatures on software security reveals that the field of defining security metrics systematically is too young to have a well acceptable definition. The problem behind the immaturity of security metrics is that the current practice of software security is still a highly diverse field and holistic and widely accepted approaches are still missing [18].

Plenty of work has been done in defining and proposing security metrics. Various security metrics exist in literature and are widely used by the security community. Most of the metrics proposed fall short of meeting the set objectives of quantifying the measures, as well as scientifically defining the same. A lot of attention has been devoted to metrics focusing on operational security of deployed systems, analyzing defect rates, known and un-patched vulnerabilities, configuration of systems.

Security metrics are hard to quantify because the discipline itself is still in the early stages of development. There is not yet a common vocabulary and not many documented best practices to follow [2]. Security metrics refer to the quantitative measurements of trust indicating how well a system meets the security requirements.

4. Security Metrics Collection

Security metrics is the measurement of the effectiveness of the organization's security efforts over time. Security metrics have always been difficult to evaluate. It helps in determining an organization whether it is secure. Several software security metrics have been proposed, and are under development, by researchers and practitioners. Some of the pertinent security metrics are listed in the following section.

Computer Viruses per Malicious Code (CVMC): This metrics counts the ratio of number of computer viruses to total number of malicious code caught: This metric measures effectiveness of automated antivirus controls [19].

Relative Attack Surface Quotient (RASQ): It is developed and used by Microsoft. This metric measures the attackability of a system, i.e., the likelihood that an attack on the system will occur and be successful. It is calculated by finding the root attack vectors, which are features of the targeted system that positively or negatively affect its security [7].

Relative Vulnerability Metric (RVM): This metric compares the calculated ratio of exploitable vulnerabilities detected in a system's software components when an intrusion prevention system (IPS) is present, against the same ratio calculated when the IPS is not present [7][20].

Security Incidents and Investigations (SII): This metrics counts the number of security incidents and investigations performed to find out such an incident. This metrics assists in monitoring security events [19].

Cost of security breaches (SBC): This metrics estimates total cost of security breaches. It gives a measure to true business loss related to security failures [19].

Time and materials (TMA): This gives measures to time and materials assigned to security functions. It presents a true business cost of running a security program [19].

Security Compliance (SC): This metrics measures compliance with security rules. It produces level of compliance matching security program goals [19].

Static Analysis Tool Effectiveness Metric (SATE): The metric combines the actual number of flaws with the tool's false positive and false negative rates, and then weights the result according to the intended audience for the resulting measurements [21].

Predictive Undiscovered Vulnerability Density Metric (UVD): This metrics is the extrapolation of Vulnerability Discovery Rate metrics. It gives measure to undiscovered or hypothetical vulnerabilities [22].

Flaw Severity and Severity-to-Complexity Metric (FSC): This metrics gives a rating reported software flaws as critical, high, medium, or low severity. It also determines whether it is possible to make a direct correlation between the number and severity of detected vulnerabilities and bugs and the complexity of the code that contains them [23].

Security Scoring Vector (S-vector) for Web Applications (SSV): This metrics is used to rate a web application's implementation against its requirements for technical capabilities, structural protection, procedural methods in order to produce an overall security score for the application [24].

Martin listed another set of metrics in his paper on software security evaluation based on a top-Down Mc Call-Like Approach [25][26].

Inalterability Metrics (IM): This metric defines the difficulty of illegal modification of the code by a potential hacker.

Physical Difficulty Metrics (PD): This metric measures the physical difficulty of code modification.

Checksum Efficiency Metrics (CE): This metrics measures the efficiency of the checksum algorithm.

Selftest Validity Metrics (SV): This metrics synthesizes an assessment on the validity of the whole selftest mechanism.

Diversity Metrics (DM): This metrics assess the diversity of code.

Number of Versions (NV): This metrics counts the different versions of the same mechanism. The code is difficult in every version, but the functionality remains the same.

Diversity Factors (DF): This metrics gives an estimate of the independence of the different versions.

Multiplicity Metrics (MM): This metric assess the number of invocations of the same mechanism. The more a mechanism is used, the more difficult it will be to circumvent.

Multiplicity Factor (MF): This metrics measures the difficulty of modification of the code implementing the mechanism.

Frequency of Use (FU): This metrics measures how often the mechanism is used.

Isolation Metrics (IM): This metrics is used to assess the isolation of the mechanism from the rest of the application and/or system.

Code Isolation (CI): This metrics assess the physical isolation of the code segment implementing the mechanism.

Data isolation (DI): This metrics addresses the data segment of the software implementing the mechanism.

Data Reuse (DR): This metrics addresses the difficulty of modifying the operational parameters of the mechanism when it is not in use.

Context Isolation (CI): This metrics address the isolation provided from the context.

Interruptibility Metrics (IM): This metrics addresses the resistance of the mechanism against interrupt driven attacks.

Mandatory Mediation (MM): This metrics is to assess of the mechanism is used every time it could.

Mediation Factor (MF): This metrics establishes the ratio between the effective use of a mechanism and its potential use.

Mediation Efficiency (ME): This metrics estimates the efficiency of use, taking into account that this efficiency is related to the situation of the mechanism in the total system.

Number of Mediation (NM): For each function using the mechanism, this metrics measures the number of times it is used.

Auditability Metrics (AM): This metrics is aimed to assess if the software leaves auditable traces of its use.

Listing of Access Denial (AD): This metrics evaluates the performance of the mechanism when it denies an access or any operation.

Alarm Triggering Metrics (AT): This metrics evaluates the performance of the alarm triggering mechanism.

Non Standard Behavior Detection (BD): This metrics aims at assessing the efficiency of such systems detecting when the behavior of a subject deviates from its standard.

Listing of Granted Access (GA): This metrics evaluates the performance of the system when it keeps tracks of the granted accesses.

5. Security Metrics Characteristics

It is inevitable facts that metrics are important to software security to measure the success of security policy, mechanism, or implementations. Metrics can be an effective tool for software security practitioners to measure the security strength and levels of their systems, products, processes, and readiness to address security issues they are facing. Metrics can also help identify system vulnerabilities, providing guidance in prioritizing corrective actions, and raising the level of security awareness within the organization [9]. Software security metrics are quantifiable, feasible to measure, and repeatable. They provide relevant trends over time and are useful in tracking performance and directing resources to initiate performance improvement actions [8].

Jelen believes that a good metric should be Specific, Measurable, Attainable, Repeatable and Time-dependent (SMART) [11]. Payne remarks that truly useful security metrics indicate the degree to which security goals such as data confidentiality are being met [18] [27][28]. Characteristics of good security metric should include the followings [14]:

- A good security metrics should be able to measure the right thing, for which it has been written;
- It should also provide quantitative measurement to make some decisions;
- It should be capable enough to be measured accurately;
- A good metrics should be validated in prior of its use;
- Metrics should be less expensive
- It should be available in early stage of software development life cycle;
- It should be able to predict overall security of software and vulnerability of software under development;
- The security metrics should be able to be refereed independently;
- It should be repeatable in nature so that the results are independent of the analyst performing the measuring;
- Good security metrics should be scalable from small single-computer systems to large nation-scale enterprise networks.
- It should generate reproducible and justifiable measurements
- It should measure something of value to the organization
- It should be able to determine real progress in security posture
- It should be capable of applying to a broad range of organizations while producing similar results
- It should help determining the order in which security controls should be applied
- It should determine the resources needed to apply to the security program

A measurement, by itself, is not a metric. Time has to be brought into the picture, and a metric alone is not the answer to all the organization's problems. The metrics have to enlighten the organization by showing some type of progress.

6. Security Metrics Development Process

Organizations that measure successes and failures of past and current security investments may use security metrics to justify and direct future security investments. It is well understood and common believe that metrics assists in improving accountability to stakeholders, ensuring an appropriate level of mission support, determining software security program effectiveness, and improving customer confidence [8]. In absence of any standard framework for identifying and developing security metrics, it appears to be advantageous to make an effort to design such a framework to carryout security metrics early in the development life cycle. The framework facilitates tailoring security metrics to a specific organization and to different stakeholders groups in each organization.

6.1. Generic Guidelines

The guidelines before following the process to develop the security metrics early in the development life cycle may be listed as follows:

- Assure compliance/ adherence to collect a generally-accepted set of characteristics that good design possesses.
- Identify and persist with all the security-specific issues involved in design phase.
- Identify policies and standards as a source of software security metrics.
- Assure to control somehow all the extraneous and intervening factors that may affect the outcome based prediction.

6.2. Premises

The following premises have been considered when the proposed framework is being used to develop a security metrics:

- There is no universally agreed-upon definition for each of high-level security factors.
- The set of security attributes used in the development of the framework has been defined operationally in the context.

- A common set of features for the desired metrics may be used to form the basis for its development.
- The recourse optimization in SDLC depends on the early use of procedure for metrics specification and uncovering of vulnerabilities as far as possible.
- The approach to risk estimate should be more applicable to identifying low security software than the highly secured code.

6.3. The Framework

The development process of the metrics is comprised of six phases together with prescriptive steps for each and has been depicted pictorially in Fig. 1 Such a framework has been proposed on the basis of integral and basic components for designing good security metrics. The first phase starts with the conceptualization. Planning for the desired metrics is treated as an important task and has been putforth as a second phase, followed by the phases termed as development, theoretical validation, experimental validation and packaging. An attempt has been made to symbolically represent the spirit of designing a security metric and make the framework prescriptive in nature followed by a brief description of each o f the phases comprising the depicted steps in the special reference to development of metrics.

Conceptualization: Conceptualization is one of the foremost tasks of any comprehensive problem-solving activity, where an initial brainstorming activity is undertaken to understand the problem, jot down ideas for solution and to realize problem-related facts. In this phase, the need and significance of the metrics to be developed is assessed. The developmental feasibility will also be checked. A strong theoretical basis will be prepared to develop such a metrics. Metrics attributes will be selected and features will be identified.

Planning: Planning assists to get success in a problem solving situation. A precisely defined plan provides guidance to the developer as it works as a roadmap. There is no doubt, that a metric will have little value if it is designed outside a well-developed structural framework. Strategic planning will be carried out for the metrics development. Security policies, guidelines and procedures are reviewed. Security factors are identified and design characteristics are explored. A link is established between identified security factors and design characteristics.

Development: Software security metrics are an integral part of the state-of-the-practice in software security engineering. Well designed metrics with documented objectives may help the organization to mitigate the vulnerabilities. Thus, designing is the most important and critical step towards the development of desired security metrics. As a subtask, stakeholders and interests are identified. Metrics program goals and objectives are defined. The metrics to be generated are decided. A metrics computation is established and finally a security metrics is formulated. Theoretical Validation: Theoretical validation of software security metrics provides the supporting evidence as to whether a measure really captures the internal attributes that it purports to measure. The main goal of theoretical validation is to assess whether a metric actually measures what it supposed to measure. A theoretical basis is examined in this phase. Experts review is conducted and observations are examined critically. On the basis of the observations made, changes are identified to be incorporated.

Empirical Validation: Testing is one of the best empirical research strategies, performed through quantitative analysis of experimental data on implementation. Hence it is necessary to place the developed security metrics under testing. A viable experiment is designed and pre-tryout is performed and reviewed. Changes are identified and tryout is performed. Result from tryout is analyzed, and as a conclusion, metrics is finalized.

Packaging: This is the conclusive phase of the metric development process. During this phase the developed metric is prepared with the needed accessories to become a ready-to-use product, like any other usable product. Metrics is introduced and its all accessories are described. A usage guideline is prescribed and a typical example is worked out. An implementation mechanism is prescribed at last.

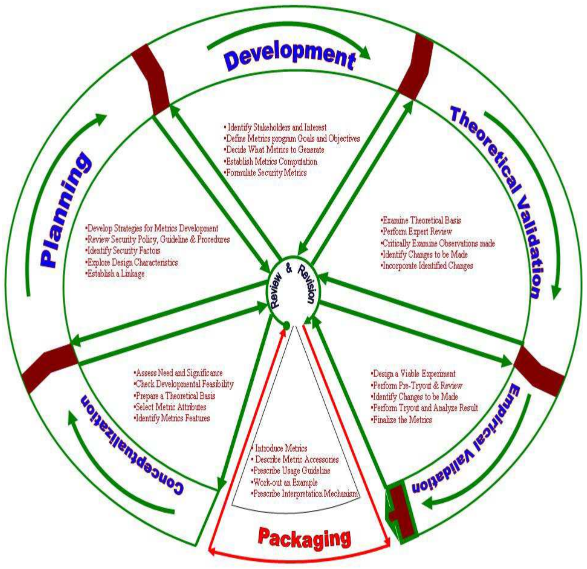
Development
• Identify Stakeholders and Interest
•Define Metrics program Goals and Objectives
•Decide What Metrics to Generate
•Establish Metrics Computation
•Formulate Security Metrics
Theoretical Validation
•Examine Theoretical Basis
•Perform Expert Review
•Critically Examine Observations made
•Identify Changes to be Made
•Incorporate Identified Changes
Planning
•Develop Strategies for Metrics Development
•Review Security Policy, Guideline & Procedures
•Identify Security Factors
•Explore Design Characteristics
•Establish a Linkage
Review & Revision
Conceptualization
•Assess Need and Significance
•Check Developmental Feasibility
•Prepare a Theoretical Basis
•Select Metric Attributes
•Identify Metrics Features
Empirical Validation
•Design a Viable Experiment
•Perform Pre-Tryout & Review
•Identify Changes to be Made
•Perform Tryout and Analyze Result
•Finalize the Metrics
Packaging
• Introduce Metrics
• Describe Metric Accessories
•Prescribe Usage Guideline
•Work-out an Example
•Prescribe Interpretation Mechanism

Fig 1. *Security Metrics Development Framework*

7. Conclusion

With this technological advancement, security is a relatively new concept for many of the organizations. Within the security realm, quantify security is still a relatively new theme. Every software system has security vulnerabilities and risks to certain degrees. It is critical for software security researchers and practitioners to identify these security risks, assessing the probability of their occurrences and the damage they could cause, and then develop security policy and mechanism to prevent or reduce the potential damages from the exploits of those security vulnerabilities. There are no well-established processes or methods to measure software security. To a great extent, different organizations have developed and deployed their own methods in measurements. Security Metrics provides a way to measure your security program. It facilitates collecting and documenting program status and reporting on the current situation and gap analysis. A framework to develop security metrics early in the development life cycle has been proposed. The proposed framework comprises of six steps including conceptualization, planning, development, theoretical validation, empirical validation and packaging. Our next step will be to implement the proposed framework in order to develop a design time security metrics.

References

[1] O. S. Saydjari, Risk: A Good System Security Measure, Proceedings of the 30th Annual International Computer Software and Applications Conference (COMPSAC'06) 0-7695-2655-1/06 $20.00, IEEE, 2006.

[2] S. Naqvi and M. Riguidel, Quantifiable Security Metrics for Large Scale Heterogeneous Systems, 1-4244-0174-7/06/$20.00, IEEE, pp. 209-215, 2006.

[3] W. Qu, D. Zhang, Security Metrics Models and Application with SVM in Information Security Management 1-4244-0973-X/07/$25.00, IEEE pp. 3234-3238, 2007.

[4] A. Ozment, Software Security Growth Modeling: Examining Vulnerabilities with Reliability Growth Models, in: Quality of Protection: Security Measurements and Metrics, Dieter Gollman, Fabio Massacci and Yautsiukhin, Artsiom.

[5] J. M. Wing, Software Security, First Joint IEEE/IFIP Symposium on Theoretical Aspects of Software Engineering (TASE'07), 0-7695-2856-2/07 $20.00, IEEE, 2007.

[6] Since Metricon 1.0, a second “mini-Metricon” was held in February 2007 at the University of San Francisco. See “Metricon 1.0” web page. securitymetrics.org [Last updated September 20, 2006, by Andrew Jaquith].

[7] ‘Software Security Assurance”, State-of-the-Art Report (SOAR) Information Assurance Technology Analysis Center (IATAC) Data and Analysis Center for Software (DACS) Joint endeavor by IATAC with DACS July 31, 2007.

[8] G. Agarwal, IT Security Metrics, 08Feb, 2008.http://cobitexpert.com/index.php?itemid=3

[9] A. J. A. Wang, Information Security Models and Metrics, 43rd ACM Southeast Conference, ACM, March 18-20 Kennesaw, GA, USA. pp. 178-184, 2005.

[10] J. Hallberg, A. Hunstad and M. Peterson, A Framework for System Security Assessment, Proceedings of the 2005 IEEE Workshop on Information Assurance and Security, United States Military Academy, West Point, NY, pp. 224-231, 2005

[11] G. Jelen, SSE-CMM Security Metrics. NIST and CSSPAB Workshop, Washington, D.C., June 2000.

[12] J. I. Alger, On Assurance, Measures, and Metrics: Definitions and Approaches. Proc. of Workshop on Information Security System Scoring and Ranking (WISSSR), ACSA and MITRE, Williamsburg, Virginia, May, 2001, proceedings published 2002.

[13] Z. Abbadi, ST13: Security Metrics: What can you test? Web Reference, 21 January, 2008.

[14] O. S. Saydjari, Is Risk a Good Security Metric? QoP’06, Alexandria, Virginia, USA. ACM 1-59593-553-3/06/0010, pp. 59-60, , October 30, 2006.

[15] ACSA (2002), Proc Workshop on information Security System Scoring and Ranking, Applied Computer Security Associates, 2002.

[16] M. Greenwald, C. Gunter, E. Knutsson, A. Sccdrov, J. Smith & S. Zdancewic, Computer Security is not a Science, Large-Scale Network Security Workshop, Landsdome, VA, 2003.

[17] Seemet, Security metrics consortium, 2004. http://www.secmet.orp

[18] Department of Homeland Security, Security in the Software Lifecycle, Making Software Development Processes—and Software Produced by Them—More Secure, DRAFT Version 1.1 - July 2006.

[19] D. A. Chapin and S. Akridge, How Can Security Be Measured? Information Systems Control Journal, Volume 2 2005.

[20] C. Cowan, Relative Vulnerability: An Empirical Assurance Metric, Presented at the 44th International Federation for Information Processing Working Group 10.4 Workshop on Measuring Assurance in Cyberspace (Monterey, CA, 25-29 June 2003).

[21] F. Stevens, Validation of an Intrusion-Tolerant Information System Using Probabilistic Modeling, MS thesis, University of Illinois, Urbana-Champaign, IL, 2004.

[22] O.H. Alhazmi, Y. K. Malaiya, and I. Ray, Security Vulnerabilities in Software Systems: a Quantitative Perspective, Proceedings of the IFIP WG 11.3 Working Conference on Data and Applications Security, Storrs, CT, August 2005.

[23] Pravir Chandra, “Code Metrics”, Presented at Metricon 1.0 (Vancouver, BC, Canada, 1 August 2006).

[24] R. R. Barton, W. J. Hery, and P. Liu, An S-vector for Web Application Security Management, working paper, Pennsylvania State University, University Park, PA, January 2004.

[25] S. Martin, Software Security Evaluation Based on a Top-Down Mc Call-Like Approach, IEEE 1988, pp. 414-418.

[26] D. B. Aredo, Metrics for Quantifying the Impacts of Monitoring on Security of Adaptive Distributed Systems, Master Thesis Proposal – II, December 2005.

[27] S. C. Payne, A Guide to Security Metrics, SANS Institute Information Security Reading Room, June 2006.

[28] R. Savola, Towards a Security Metrics Taxonomy for the Information and Communication Technology Industry, International Conference on Software Engineering Advances(ICSEA 2007) 0-7695-2937-2/07,2007, IEEE.