ISSN:0975-9646

A Review of Security Metrics in Software Development Process

Smriti Jain[#], Maya Ingle[*]

[#]*MCA Department, SRGPGPI*
[*]*School of Computer Science, DAVV*
Indore, M.P., India

***Abstract* — Security level, security performance, and security indicators have become standard terms to define security metrics. The data derived from these metrics helps in measurement of software security. The metrics help achieve security objectives – confidentiality, integrity and availability. The security can be assessed for further improvement during development process of the software or the product itself. The security assessment is helpful for software developers, risk management team, executives of the company, etc. Our paper reviews both the kinds of metrics and confers the results.**

***Keywords*— Security Metrics, Software Development Process.**

I. INTRODUCTION

Software development is a complex task that involves a number of stages such as inception, initial design, detailed design and development, implementation and testing, operation, upkeep and retirement. Earlier, it was a trend to develop software that meets the functional needs of the customers in organizations. Whereas, in present scenario computers manage the complete working and business of an organization, which are having various branches across the globe. In the interconnected electronic world, softwares are prone to various kinds of attacks which raised the need for securing the software. Any flaw contained in the software may create the environment vulnerable. The growing number of attacks has forced the organizations to incorporate the aspect of security during development process rather than considering during post development phase. It has been observed that 90% of vulnerabilities occur as a result of flaws in design and coding [1]. Security considerations when considered as part of Software Development Process (SDP), results in a more secured product [2]. Microsoft (Security Development Lifecycle), Cigital (Touchpoint Process) and OWASP (Comprehensive Lightweight Application Security Process) are major players to utilize secured software development process. These models address good (but are not generalized) software engineering practices. The models focus on positive practices to include security in a software product, and process maturity [3]. Although, these models not generalized and are difficult to use.

The security requirements when given importance as functional requirements, needs to be measured. Metrics help the project management team to effectively manage the product as well as the software development process. Metrics aid in analysis and early detection and correction of the functionalities of the software. The metrics can assess the security risks more efficiently when considered during software development process. One of the approaches to develop security metrics is Goal/ Question/ Metric (GQM) approach. A set of metrics have been developed using GQM approach to assess security risks throughout the various stages of SDP [4]. Center for Internet Security (CIS) provided a set of security metrics related to risk management wrapping aspects of business functions. The categorization done is Incident Management, Vulnerability Management, Patch Management, Application Security, Configuration Management and Financial Metrics. The metrics are also organized on the basis of purpose and audience viz. management metrics, operational metrics, and technical metrics [5]. Research report of CERT by Software Engineering Institute presents a paper on measuring software security. The paper gives a set of security metrics that are considered during the software development process (SDP) viz. requirements engineering, architecture and design, coding and testing. Some of the metrics defined are percent of relevant security principles reflected in requirements specification, Percentage of architectural/ design components subject to attack surface analysis and measurement, Percentage of software components subject to static and dynamic code analysis against known vulnerabilities and weaknesses, Percentage of defects discovered during testing that was injected in coding; in architecture and design; in requirements specification etc. [6]. The code metrics are also defined as the representative weakness of software i.e. weaknesses that lead most vulnerabilities to be exploited by the attackers [1]. Various existing standards like Common Criteria, ISO/ IEC 27004, NIST 800-55, etc. define security metrics and are too broad to provide precise security definitions and are not able to cover all security aspects [16]. Other than the metrics defined by standards, Mellado enumerates security metrics for object oriented class diagrams, Security estimation framework, attack surface of a system, Common Weakness Enumeration (CWE), Common Vulnerability Scoring System (CVSS) and Common Misuse Scoring System (CMSS). The various security metrics are discussed and compared on the basis of security characteristics like authenticity, confidentiality, conformance, detection of attacks, availability, integrity, non-repudiation, traceability etc. in [7]. Literature review reveals that the most of the relevant security metrics either focus on assessing security of software on system level perspective or regarding inclusion of security metrics during software development stages. But this realization is not at a glance. However, it is possible to judge the scope to develop security metrics.

The security of software can be monitored systematically using security metrics. We first discuss the software development process along with the importance of security at its various stages in Section 2. Section 3 presents a review of various security metrics in software development process as well as system level security metrics. Finally, we conclude with results in Section 4.

II. Security in SDP

To understand the significance of security, we discuss the software development process and the importance of security aspect in the SDP.

A. Software Development Process

A number of general software development models are in practice like waterfall model, prototype model, RAD, incremental model, iterative model etc. A typical software development process begins with requirements gathering, analysis, design, coding and implementation, testing, and maintenance. The system requirements stage emphasizes on gathering requirements from customers. The requirements mainly relate to software functionality and the quality attributes including Non-Functional Requirements (NFR). After gathering requirements, analysis of the scope of development is determined. What and why of requirements is converted to design in the next stage. The design mainly illustrates the major components of the software and their relationships. The design can be detailed to even take care of NFRs. The design elements includes functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudocode, and a complete entity-relationship diagram with a full data dictionary that describes software in sufficient detail so that the programmers may develop the software easily [8]. Coding implements the design specification. Developers also perform unit and module testing and further integrates the system. Testing is the final step towards all unaddressed issues of the previous stages. It reports the errors, verifies if the system meets the requirements, the design, and expected work. Testing judges the quality of the product and identifies the risks associated. Once the software is deployed, new users need to be trained. Lack of training increases the chances of not adopting the software. Maintenance phase includes correcting faults, adapting environment, changing and enhancing the delivered and installed product. Maintenance also helps to cope up with newly discovered problems.

B. Importance of Security

With increased connectivity, security has become one of the very important considerations of software as the systems are becoming more complex in nature. Although security is critical to many domains like banking, airline, national defence etc., experience of organizations show that security is still not given due importance during software development process. A study had shown that 47% of banks place secure login boxes and 55% put contact information and security advice on insecure pages [9]. To implement security in software development process, firstly sensitivity of information should be analysed. This will help in taking decisions regarding security needs of the organization. Security can be implemented during SDLC by considering all phases of software development starting from requirements gathering to maintenance. There are number of security issues at every stage of software development. A widespread cause of defect in a product is requirements gap. This gap is result of lack of knowledge regarding security issues at the business level as well as lack of understanding of attack scenarios. The security requirements if understood well can lead to secured product. Similarly, the insecure design is result of not considering security as prime objective, and also due to lack of knowledge of security principles, guidelines and attack patterns [8].

III. Security metrics – A review

We discuss existing security metrics in SDP as well as at system level in this section.

A. Security Metrics in SDP

A number of security metrics have been specified which are further explained that portray the security issues of different software development stages.

Requirements Gathering and Analysis **-** The security can be assessed during requirements phase using metrics like Total number of security requirement (Nsr), Ratio of security requirements (Rsr), Number of omitted security requirements (Nosr) and Ratio of the number of omitted security requirements (Rosr) [4]. The measures for requirements engineering phase also include Percent of relevant software security principles reflected in requirements specifications, Percent of security requirements that have been subject to analysis, and Percentage of security requirements covered by attack patterns, misuse/ abuse cases, and other specified means of threat modelling and analysis [6].

Software Design **-** The design metrics established include Number of design decisions related to security (Ndd), Ratio of design decisions (Rdd), Number of security algorithms (Nsa), Number of design flaws related to security (Nsfd), and Ratio of design flaws related to security (Rfd) [4]. Architecture and design metrics include Percentage of architectural/ design components subject to attack surface analysis and measurement, Percentage of architectural/ design components subject to architectural risk analysis, and Percentage of high-value security controls covered by security design patterns [6]. A set of metrics have been derived from view point of information flow based on object-oriented design artefacts viz. composition, coupling, extensibility, inheritance and design size of the given object-oriented, multi-class program.

CC is a set of critical classes in a design D and CP is a set of the composed-part critical classes in the same design. The Composite-Part Critical Classes metric is given as

$$CPCC(D) = 1 - \left(\frac{|CP|}{|CC|}\right)$$

A set of classes, C in a design D, set of classified attributes as CA_j, and $\alpha(CA_j)$ is the number of classes which may interact with classified attribute CA_j. The Critical Class Coupling, CCC metric for design D is expressed as

$$CCC(D) = \frac{\sum_{j=1}^{ca} \alpha(CA_j)}{(|C|-1) \times |CA|}$$

The Classified Methods Extensibility metric is defined as

$$CME(D) = \frac{|ECM|}{|CM|}$$

Where, CM is a set of classified methods in a design D and ECM is a set of extensible classified methods in the same design such that ECM $\subseteq$ CM.

Metrics are defined for Inheritance for Critical Superclasses Proportion (CSP), Critical Superclasses Inheritance (CSI), Classified Methods Inheritance (CMI), and Classified Attributes Inheritance (CAI).

A set of Critical Classes (CC) and Critical Superclasses in the same hierarchy (CSC) such that CSC $\subseteq$ CC. CSP is defined as

$$CSP(H) = \frac{|CSC|}{|CC|}$$

A set of classified methods (CM) in hierarchy H, the classified methods inherited (MI) in the same hierarchy such that $MI \subseteq CM$, then CMI metric is given as

$$CMI(H) = \frac{|MI|}{|CM|}$$

A set of Classes, Cj for Hierarchy, H, a set of Critical Superclasses (CSC_k) in same hierarchy, $CSC \subseteq CC$ and $CC \subseteq C$. Let $\beta(CSC_k)$ be the number of classes which may inherit from superclass CSC_k, then CSI is defined as

$$CSI(H) = \frac{\sum_{k=1}^{csc} \beta(CSC_k)}{(|C|-1) \times |CC|}$$

CA be the set of classified attributes in Hierarchy H, classified attributes inherited (AI) in the same hierarchy, and $AI \subseteq CA$, then CAI metrics is given as

$$CAI(H) = \frac{|AI|}{|CA|}$$

Lastly, the Critical Design Proportion Metric is given as

$$CDP(D) = \frac{|CC|}{|C|}$$

Where, C is set of classes in design D, CC is critical classes in same design such that $CC \subseteq C$ [11].

Coding/ Implementation – A number of coding/ implementation metrics recognized consist of Number of implementation errors found in the system (Nerr), Number of implementation errors related to security (Nserr), ratio of implementation errors that have impact on security (Rserr), Number of exceptions that have been implemented to handle failures related to security (Nex), Number of omitted exceptions for handling execution failures related to security (Noex), and Ratio of the number of omitted exceptions (Roex) [4]. Security measures for coding involve Percentage of software components subject to static and dynamic code analysis against known vulnerabilities and weaknesses, Percentage of defects discovered during coding that was injected in architecture and design in requirements specification, and Percentage of software components subject to code integrity and handling procedures, such as chain of custody verification, anti-tampering, and code signing [6]. The metrics are also defined at the source code level like Stall Ratio, Coupling Corruption Propagation (CCP), Critical Element Ratio (CER). Stall ratio measures program's progress as hindered by vivacious activities. It is calculated as the ration of Lines of non-progressive statements in a loop to Total lines in the loop. The good stall statements include statements to write error messages, writing logs etc. Code with high stall ratio is more prone to attack. CCP measures the total number of methods that could be affected by erroneous originating method. It is given as number of child methods invoked with the parameter(s) based on the parameter(s) of the original invocation. CER is calculated as ratio of Critical Data Elements in an Object to Total Number of Elements in the Object. The critical elements can be corrupted by the malicious user input. If the critical data objects change, the whole process may be subject to security risk. Thus, the code with higher CER should be tested more carefully [10].

Testing - Testing phase relates to metrics like Ratio of security test cases (Rtc) and, Ratio of security test cases that fail (Rtcp) [4]. Measures for testing security also comprise of Percentage of defects discovered during testing that was injected in coding, in architecture and design, and in requirements specification; Percentage of software components with demonstrated satisfaction of security requirements as represented by a range of testing approaches; and Percentage of software components that demonstrated required levels of attack resistance and resilience when subject to attack patterns, misuse/abuse cases, and other specified means of threat modelling and analysis [6]. SANS reading room provides testing metrics as Security Testing Coverage [15].

Maintenance – The observed maintenance phase metrics include Ratio of software changes due to security consideration (Rsc), and Ratio of patches issued to address security vulnerabilities (Rp) [4]. Other metrics are Mean time between security incidents, Mean-time to patch, Mean-time to complete changes, Percent of changes with security exceptions [5].

Documentation – Technical documentation has been assessed for quality using GQM approach using clone detection and test coverage analysis [12].

B. System Level Security Metrics

Security metrics are defined on the basis of vulnerabilities and are proposed on the basis of Common Vulnerabilities and Exposures (CVE), an industry standard for vulnerability and exposure names, and the Common Vulnerability Scoring System (CVSS), a vulnerability scoring system designed to provide an open and standardized method for rating software vulnerabilities. Metrics defined is representative weakness of software i.e. those weaknesses that lead most of the vulnerabilities to be exploited by the attackers and are given below

$$SM(s) = \sum_{n=1}^{m} \left(P_n \times W_n\right), \qquad (1)$$

Where SM(s) is security metrics for the software s, and Wi (i = 1, 2, …, m) are the severity of weakness in the software s and P (i = 1, 2, …, m) represents the risk of the corresponding weakness. W_n is the severity of the weakness and is given by

$$W_n = \frac{\sum_{i=1}^{K} V_i}{K} \qquad (2)$$

where K is number of vulnerabilities of weakness W with corresponding base scores as V_i.

$$P_n = \frac{R_n}{\sum_{i=1}^{m} R_i} \qquad (3)$$

P_n is expressed as percentage and is given by occurrence of each weakness in the overall weakness

$$R_n = \frac{K}{M} \qquad (4)$$

R_n is the frequency of occurrences of each representative weakness, K during M months.

$$\sum_{n=1}^{m} P_n = 1 \qquad (5)$$

Thus, the authors have defined software security metrics on the basis of representative weakness [1].

Common Criteria (CC) also defines seven assurance levels varying from EAL1 to EAL7, and are metrics to rank assurance on evaluated products [13]. NIST presents system level controls addressing the information security program which includes access control, awareness and training, audit and accountability, etc. It provides some the measures for audit processing failures that include software/ hardware errors, failures in the audit capturing mechanisms, and audit storage capacity being reached or exceeded [14]. CIS security outcome and practice metrics define 20 metrics under six business functions. Risk Assessment Coverage and Security Testing Coverage metrics assess the application security. The change management metrics include Mean time to complete changes, Percent of Changes with security reviews, Percent of changes with security exceptions. Vulnerability management metrics include Mean-time to mitigate vulnerabilities, Number of known vulnerability incidences etc. [15].

IV. Results and Conclusion

A number of metrics listed in the Section 3 are established for different software development life cycle activities. It has been revealed that most of the security metrics in software development process assess security risks and evaluate risk coverage at each stage of software development. Some of the authors do not provide the means for data collection and the method to calculate the metrics. Some of the metrics developed had not been tested for assessing the impact on product security, while others have proposed security metrics for specific life cycle activity. In the paper that defines security metrics for risk assessment during various stages of life cycle, some of the metrics can be assessed only later in the software development process and not during the software development stage itself. Another set of metrics proposed on code inspections enable to assess security after full system implementation which makes it impossible to fix them early. A set of metrics for object oriented design assesses security when UML class diagram using UMLsec and SPARK's annotations are provided, limiting its utility. The code metrics developed are based on the vulnerabilities as listed in CWE, CVE and CVSS that makes it quite useful for assessing only the code.

It is observed that the metrics developed have not considered the reasons for insecurity during SDP. Thus, there remains the scope of development of metrics for quantitative assessment of security using the reasons for security loop holes in the software identified during SDP.

References

[1] J. A. Wang, H. Wang, M. Guo, and M. Xia, "Security Metrics for Software Systems," In the Proc. of ACMSE '09 March 19-21, 2009.

[2] Network magazine, Oct 2006,. [Online] Avaliable: http://www.networkmagazineindia.com/200610/vendorvoice02.shtml

[3] *Secure SDLC's Compared*, KRvW Associates, LLC, 2008. [Online] Avaliable: http://www.secappdev.org/handouts/2008/Secure%20SDLCs%20compared.pdf

[4] K. Sultan, A. En-Nouaary, A. Hamou-Lhadj, "Catalog of Metrics for Assessing Security Risks of Software throughout the Software Development Life Cycle." In the Proc. of *International Conference on Information Security and Assurance*, IEEE Computer Society, 2008, pp. 461-465.

[5] *The CIS Security Metrics*, Nov. 2010. Accessed on 18-Oct-2011. [Online]. Available:

https://benchmarks.cisecurity.org/tools2/metrics/CIS_Security_Metrics_v1.1.0.pdf

[6] J. Allen, "Measuring Software Security," CERT Research Annual Report, Software Engineering Institute, Carnegie Mellon University, pp. 64-65, 2009.

[7] D.Mellado, E. Fernández-Medina and M. Piattini, "A Comparison of Software Design Security Metrics," In the proc. of ECSA, 2010, pp 236-242.

[8] S. Jain, "Involving Security in Software Development Process – A Suggestive View". In the Proc. of *National Conference on Emerging Technologies in Electronics, Mechanical and Computer Engineering*, Indore, India, 2010.

[9] "Potentially Serious Security Flaws Found In Most Bank Websites, Including Large Bank Sites, Study Shows." *Science Daily*, July 23, 2008. http://www.sciencedaily.com/releases/2008/07/080722175802.htm.

[10] I. Chowdhury, B. Chan, and M. Zulkernine, "Security Metrics for Source Code Structures," In the Proc. of SESS'08, Germany, May 17–18, pp. 57-64, 2008.

[11] B. Alshammari, C. Fidge and D. Corney, "Security Metrics for Object-Oriented Designs," IEEE Computer Society, pp. 55-64, 2010.

[12] A. Wingkvist, M. Ericsson, R. Lincke and W. Lowe, "A Metrics-Based Approach to Technical Documentation Quality," In the Proc. of Seventh International Conference on the Quality of Information and Communications Technology, IEEE Computer Society, 2010.

[13] CCIMB-2004-01-003, *Common Criteria for Information Technology Security Evaluation: Security assurance requirements and Protection Profiles* Version2.2, 2004.

[14] *Information Security*, NIST Special Publication 800-53, Revision 3, August 2009. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/800-53-Rev3/sp800-53-rev3-final.pdf

[15] C. E. Nelson, "Security Metrics - An Overview", *ISSA Journal*, pp. 12-18, , August 2010.

[16] A.J.A. Wang, "Information Security Models and Metrics," 43rd ACM Southeast Conference, pp. 2.178 – 2.184, 2005.