International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868
Foundation of Computer Science FCS, New York, USA
Volume 12 – No. 6, August 2017 – www.ijais.org

Significance of Security Metrics in Secure Software Development

Shams Tabrez Siddiqui
Department of Computer Science,
College of Computer Science & Information Systems
Jazan University, KSA

ABSTRACT

With increasing advancement of technology in the past years rise various security issues and problems. In this connected world, security is a paramount and challenging issue in software development and is the demand of time. However usually engineers/developers think about it after the development of the entire software and at that it's too late. Though, the software developers are aware of the importance of security and its priority throughout software development life cycle. Considering the security challenging issues right from the early stages of software development and incorporating it during software development indicates good research and development.

When the metrics considered during software development process from the initial stage then it assess the security risks more efficiently. One of the best known approaches to develop security metrics is Goal/Question/Metric (GQM) approach that assesses the security risks in various stages of software development process. Software security can be measured with the help of metrics derived from the source available.

The main aim of this paper is to focus on numerous security metrics of software development phases and some standardized criteria is used for validation. Each and every phase have different metrics as compared to other. Those metrics are defined on the bases of their results and products. The final product derived from the proposed security metrics of the software will be secure and qualified.

General Terms

Security, software development phases, validation.

Keywords

Security risks, software development life cycle, metrics, GQM.

1. INTRODUCTION

Rapid growth of technology in last few decades increases the security related issues and now most of the researchers are considering security problems seriously. Nowadays security metrics are used in numerous fields and considering security problems, issues and challenges from the early stage of software development life cycle indicates the good research and development [1]. Higher priority should be given to the security related issues from the initial stage of the software development. Secure software cannot deliberately force to fail and remains correct and inevitable in spite of intentional efforts. Regularizing software security metrics assures the quality and security of the entire system. A secure system avoids service failure and measures availability, reliability, and maintenance of the system. Secure system does what it is supposed to do and what is not supposed to do [2].

The enlisted aspects namely; availability, reliability, and maintenance of the system are considered as secure software. Security usually considered by the developers as a post development activity. Most of the organizations, developers or software engineers try to incorporate security as a patch after software development, but security is not a feature, it is an emergent property of a complete system [3]. Security will be more effective and efficient if it is considered during pre-development and development phases [1] [3]. Therefore, security should be incorporated in each and almost every phase from the initial stage of the software development phase. Most of the organizations spend a huge amount on purchasing firewalls and antivirus for the software, even though there software's are not secure [4]. Due to exploitation of security flaws, they incur significant losses of data and information in the organizations. Metrics are virtuous if it is clearly specified, measurable, time dependent, repeatable and understandable [5] [6].

This paper discusses numerous security metrics in almost all software development phases and some standardized criteria is been used for validation. Almost all the phase of the software development lifecycle has different security metrics as compared to other that are defined on the basis of their results and products [6]. Using proposed security metrics during software development cycle for secure and qualified final product. The standard security metrics is used for security measurement. In architecture design and secure operations the security metrics are most significant factors. These security metrics can be used effectively in quality and security assurance applications [7] [8].

2. SECURITY METRICS

Security metrics can be defined as a standard terms as security level, security indicators and security performance. Software security can be measured with the help of the metrics derived from the data or information of the software which is under development phase.

It is clearly known that, when the metrics considered during the initial phase of the software development process will assess the security risks more effectively and efficiently. Goal/Question/Metric (GQM) approach is one of the best known approaches to develop security metrics and to assess the security risks in the stages of software development process [9] [10].

2.1 Software Security Metrics

- Software measures are troublesome (LOC, FPs, Complexity etc.)

- Metrics are context sensitive and environment-dependent
- Aggregation may not lead to strength
- Architecture dependent

3. SECURITY METRICS IN SOFTWARE DEVELOPMENT PROCESS

A number of security metrics have been specified and described in detail that portrays the security related issues in the development stages of the software also given in table 2 [9] [10].

3.1 Pre-Requirement Phase

Security activities performed in pre initial phase set the foundation for all the activities starting from the initial stage to final stage such as requirements phase to the maintenance phase [1] [12] [20]. The activities performed during pre-requirement phase are:

- **Is security required in the system:** Before assessing security requirement the study of the system and its requirement should be done?
- **Is security possible for the system:** Before considering security metrics we have to check whether security requirement is possible for the system or not.
- **Number of possible security requirements in each phases of Software Development Process**
- **Total Number of security requirement in Software Development Process**

3.2 Requirement Gathering and Analysis Phase

In this phase, software engineers assess security requirements and evaluate it to consider properly. The metrics that makes this phase more precise are [10]:

- **Number of priority security requirements (Npsr):** This metrics is for the number of security requirements that have more priority than the others and consider the requirements which due to attacks on the system will affect or destroy the system most.
- **Number of least priority security requirements (Nlsr):** Number of security requirements that have less priority than the other and the considered requirements will not effect the system much or their will be no effect on it.
- **Total number of security requirements (Nsr):** This metrics assess the total number of security requirements (priority and least priority security requirements) identified through analysis phase of the software development.

  Nsr = Npsr + Nlsr

- **Ratio of security requirements (Rsr):** Rsr can be calculated as:

  $Rsr = |SR| / |R|$
  R: Systems all requirements set.
  SR: Systems security requirements set.
  Even SR is said to be a subset of R.

- **Number of omitted security requirements (Nosr):** Number of security requirements that have been omitted or not considered due to any reason, but have a high risk and possible impact of severe attacks on the system.
- **Ratio of the number of omitted security requirements (Rosr):** Rosr defined as :

  Rosr = Nosr/ (Nosr + Nsr)

3.3 Software Design Phase

In this phase, software engineers assess security requirements and evaluate it to deal properly during design phase or not. Metrics for design phase are [10]:

- **Number of design decisions related to security (Ndd):** This security design metrics considers the number of design decision that addresses the security requirement of the system.
- **Ratio of design decisions (Rdd):** The purpose of this metrics is to measures the ratio of design decisions related to security of the system.

  Rdd = Ndd / Nd

  Nd is the total number of design decisions

  of the complete system.

- **Number of security algorithms (Nsa):** This metrics measures the number of security algorithms found in the entire system.
- **Number of design flaws related to security (Nsdf):** The aim of this security design flaws metrics is to ruminate the number of security related design flaws occurs due to improper planning with improper consideration of security requirements principles.
- **Ratio of design flaws related to security (Rdf):** This metrics measures the ratio of design flaws related security metrics of the system, considering number of design flaws that are related to security over design flaws relevant to the complete system.

  Rdf = Nsdf / Ndf

  Ndf is the number of design flaws pertinent to the complete system.

3.4 Coding Phase

In this phase the metrics measure source code quality properties that enhance program security are given below [11]:

- **Stall ratio (Sr):** This measures, the delay of program progress by frolicsome activities. In a program, there are certain statements which may not be severely essential for making the program progress towards its desired goal. But in the program there may be certain statements that do not confer to the overall progress of the program [11] [12]. The metrics given below measures the progress of the program:

$$Stall\ Ratio = \frac{Lines\ of\ non\ progressive\ statements\ in\ a\ loop}{Total\ lines\ in\ a\ loop}$$

- **Critical element ratio (Cer):** A security risk appears if certain necessary data objects are altered that may threaten the process as a whole. Henceforth, the more critical elements in a class, the higher are the security risk [11] [12]. This risk enumerated in the following way:

$$Critical\ elements\ ratio = \frac{Critical\ Data\ Elements\ in\ an\ Object}{Total\ number\ of\ the\ Elements\ in\ the\ Object}$$

- **Coupling corruption propagation (Ccp):** The metrics coupling corruption propagation is destined to assess the total number of methods that could be affected by flawed instigating methods. Potential security flaw fallouts when a critical parameter is imposed to remain at a certain value, and the fallouts remains the same as there is no matter what other parameters are altered [11] [12]. The formal definition of this proposed metric is:

$$Coupling\ Corruption\ Propagation = \frac{Number\ of\ child\ method\ invoked\ with\ the\ parameter(s)}{based\ on\ the\ parameter(s) the\ original\ invocation}$$

3.5 Implementation Phase

The implementation metrics are given below:

- **At the time of implementation; Number of errors found in the system (Nerr):** This metrics measures the number of implementation errors of the system [10] [13]. This metric is a baseline metric and is used for defining the remaining metrics.

- **Number of implementation errors associated to security (Nserr) of the system:** This metrics is used for measuring the number of errors at the time of implementation which has the direct impact of security for the system [13] [14].

- **Ratio of implementation errors that have impact on security (Rserr):** This metrics is for finding the ratio of errors that have impact on security and can be calculated by finding the number of errors at the time of implementation related to the errors only associated to security [13].

  Rserr=Nserr / Nerr

- **Number of exceptions implemented to handle failure related to security (Nex) of the system:** This metric measures the number of exceptions which are included in the code to handle possible failures of the system due to an error that has impact on security [10].

- **Number of omitted exceptions for handling execution failures related to security (Noex):** This metrics deals with the number of missing exceptions which are omitted by developers while implementing the system [10].

- **Ratio of the number of omitted exceptions (Roex):** The ratio of the omitted exception measurement metric is:

  Roex = Noex / (Noex + Nex)

3.6 Testing Phase

In this phase the following metrics are used to test the security of the system [10].

- **Total number of security test cases (Ttc) of the system:** This is the total number of security related all the tests of complete system.

- **Number of security test cases of the system that fails (Ntcp):** This is for number of security errors, faults and failure of the system while implementing.

- **Ratio of security test cases (Rtc):** This metrics helps developers to determine the ratio of testing for security that the system has undergone.

  $Rtc = |TS| / |T|$
  T: Test cases of the systems set.
  TS: Test cases that address security issues set.

- **Ratio of security test cases that fails (Rtcp):** The aim of the metrics is to detect implementation error

  $Rtcp = |TF| / (|TP| + |TF|)$
  TP: Is a set of security associated test cases of the system that passes.
  TF: Is a set of security associated test cases of the system that fails.

3.7 Maintenance Phase

This metrics considers security during evolution and maintenance phase.

- **Ratio of software changes due to security consideration (Rsc):** In order to keep the application secure this metrics helps in identify the extent of work performed on the system [10].

  Rsc = Nsc / Nc

  Nc: Is the number of changes of complete system.

  Nsc: It is the number of changes that

  triggered via a new security requirements.

- **Ratio of patches issued to address security vulnerabilities of the system (Rp):** This metrics measures the ratio of patches that are issued to address security vulnerabilities that can be calculated [10].

  Rp = Nsp / Np
  Np: The number of patches of the entire system.
  Nsp: The number of patches related to security of the system.

4. COMPARISION OF SECURITY ACTIVITY WITH SECURITY METRICS OF THE SOFTWARE DEVELOPMENT PHASES

Security metrics can be defined as a standard terms as security level, security indicators and security performance. Software security can be measured with the help of the metrics derived from the data [1].

Table 1 shows the security activities of the software development phases whereas table 2 shows the security metrics of each phases of the software development. Security activities of each phase are important and calculating total security related issues of each phases of the software development is also crucial.

Table 1: Security Activities for Secure Software Development [10] [15-19] [21-23].

SOFTWARE DEVELOPMENT PHASES	SECURITY ACTIVITIES
Pre-Requirements Phase	Security Training, Plan and Develop Framework for Risk Management
Requirements Phase	Identify Security Requirements, Develop Use Cases, Develop Misuse Cases, Develop Security Use Cases, Documentation of Requirements
Design Phase	Build Security Architecture, Identify Interaction Points, Assets and their Access Points, Minimize Software Attack Surface, Describe Threat Models
Implementation Phase	Write Secure Code, Static Analysis and Review of Code
Testing Phase	Security Test Planning, Security Testing
Release and Deployment Phase	Security Review, Security Audit, Security Deployment

Table 2: Security Metrics in Software Development Process [10] [13-14]

SOFTWARE DEVELOPMENT PHASES	SECURITY METRICS
Pre requirement	Is security possible for the system, Is security required in the system, Number of possible security requirements in each phases of Software Development Process, Total Number of security requirement in Software Development Process
Requirement Gathering and Analysis	Number of priority security requirements (Npsr), Number of least priority security requirements (Nlsr), Total number of security requirement (Nsr), Number of omitted security requirements (Nosr), Ratio of security requirement (Rsr), and Ratio of the number of omitted security requirements (Rosr)
Software Design	Number of design decisions related to security (Ndd), Number of security algorithms (Nsa), Ratio of design decisions (Rdd), Number of design flaws related to security (Nsfd), and Ratio of design flaws related to security (Rfd)
Coding	Stall ratio (Sr), Critical element ratio (Cer),Coupling corruption propagation (Ccp)
Implementation	Number of implementation errors found in the system (Nerr), Number of implementation errors associated to security (Nserr), Ratio of implementation errors that have impact on security (Rserr), Number of exceptions that have been implemented to handle failure 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)
Testing	Total number of security test cases (Ttc), Number of security test cases that fails (Ntcp), Ratio of security test cases (Rtc) and Ratio of security test cases that fail (Rtcp)
Maintenance	Ratio of software changes due to security consideration (Rsc), and Ratio of patches issued to address security vulnerabilities (Rp)
Documentation	Technical documentation using GQM approach for quality assessment.

Table 3 exemplifies the software projects in percentages that failed or were over budget in the years (1994, 1998, 2000, 2004, 2006, 2008, 2010 and 2012). From the Table 3 it seems that the crisis in SE due to security is also one of the challenging issues. The results indicate that there is still work to be done around achieving successful outcomes from software development projects [24]. Software fails due to security reasons is also included in Chaos report.

Table 3: Chaos Reports for Data of Software Projects [25] [26] [27] [28]

Project cancelled or failed DUE to security and other reasons (in percentage)	Project Over Budget (in percentage)	Report Year and Reference
31.1%	52.7%	1994
28%	46%	1998
23%	49%	2000

18%	53% (43% for small and medium projects 82% for large projects)	2004
19%	46%	2006
24%	44%	2008
21%	42%	2010
18% (4% small projects and 38% large projects)	43% (20% for small and medium projects 52% for large projects)	2012

Table 4 summarizes the outcomes of projects over the last five years. Success factors on time, within budget with a satisfactory result are shown below in the table 4. Most of the projects that deal the security issues and perspective are challenged projects.

Table 4: Chaos Reports success factors (on time, on budget with a satisfactory result) [24].

YEAR	SUCCESSFUL	CHALLENGED	FAILED
2011	29%	49%	22%
2012	27%	56%	17%
2013	31%	50%	19%
2014	28%	55%	17%
2015	29%	52%	19%

5. CONCLUSION

The aim of this paper is to focus on the security metrics of each phase of the secure software development activities. Careful consideration of security is required right from the initial stage of the software development phases namely; pre-requirements, requirements, design to the final stages of the secure software development phases namely; implementation, testing, deployment and maintenance. The main aim of this paper is to focus and propose some security metrics that can be used to assess and avoid the risks at different stages of the software development processes. The proposed metrics for the security of the system is considered as the framework and should be incorporated right from the initial stage of the software development process.

In this paper number of security metrics have been itemized, specified and described in detail that portrays the security related issues in software development stages. The given security metrics calculates, number of possible security requirements, total number of security requirements from all the phases of software development process, omitted security requirements, priority security requirements, design decisions related to security, design flaws related to security, number of implementation errors found in the system, implementation errors associated to security, total number of security test cases etc.

The metrics also calculates; ratio of security requirement, implementation errors that have impact on security, ratio of omitted exceptions, patches issued to address security vulnerabilities, ratio of software changes due to security consideration, etc.

6. REFERENCES

[1] S. T. Siddiqui, H. S. A. Hamatta and M. U. Bokhari, “Multilevel Security Spiral (MSS) Model: NOVEL Approach”, International Journal of Computer Applications, vol. 65, no. 20, pp. 15-20, 2013.

[2] G. McGraw, “Software Security”, IEEE Security & Privacy, vol. 2, no. 2, pp. 80-83, 2004.

[3] D. G. Firesmith, “Specifying reusable security requirements”, Journal of Object Technology, vol. 3, no. 1, pp. 61-75, 2004.

[4] Dustin E, “The Secure Software Development Lifecycle”, Dev Source (sponsored by Microsoft), 2006.

[5] A. Abdi, “Using Security Metrics in Software Quality Assurance Process”, On 6th International Symposium on Telecommunications (IST'2012) IEEE, 2013.

[6] M. U. Bokhari and S. T. Siddiqui, “Metrics for Requirement Engineering and Automated Requirement Tools”, In Proceedings of the 5th National Conference; INDIACom-2011, New Delhi, 2011.

[7] M. U. Bokhari and S. T. Siddiqui, “A Comparative Study of Software Requirements Tools for Secure Software Development”, BVICAM‟S International Journal of IT (BIJIT), vol. 2, no. 2, pp. 207-216, 2010.

[8] Website: Https://techbeacon.com/9-metrics-can-makedifference-todays-software-development-teams. Accessed: January 27, 2017.

[9] M. A. Hadavi, H. M. Sangchi, V. S. Hamishagi and H. Shirazi, “Software security; A vulnerability-activity revisit”, In Proceedings of the 2008 Third International Conference on Availability, Reliability and Security (ARES’08), pp. 866-872, 2008.

[10] K. Sultan, A. E-Nouaary and A. H-Lhadj, “Catalog of metrics for assessing security risks of software throughout the software development life cycle”, In Proceeding of International Conference on Information Security and Assurance, IEEE Computer Society, pp. 461-465, 2008.

[11] 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 (ICSE’08), Leipzig, Germany: ACM, pp. 57-64, 2008.

[12] S. T. Siddiqui, “Comparative Study and Design of Software Requirement Tools for Secure Software Development”, Ph. D Thesis, Department of Computer Science, A. M. U., Aligarh, India, 2015.

[13] S. Jain and M. Ingle, “Review of security metrics in software development process”, International Journal of Computer Science and Information Technologies, vol. 2, no. 6, pp. 2627-2631, 2011.

[14] G. Caldiera, V. R. Basili and H. D. Rombach, “The goal question metric approach”, Encyclopedia of software engineering, J. J. Marciniak(ed.), New York, USA: John Wiley & Sons, vol. 2, pp. 528-532, 1994.

[15] S. R. Ahmed, “Secure software development - Identification of security activities and their integration in software development lifecycle”, Master’s Thesis, School of Engineering, Blekinge Institute of Technology, Sweden, 2007.

[16] B.W. Boehm, “A spiral model of software development and enhancement”, TRW Defense Systems Group, IEEE Computer, vol. 21, no. 5, pp. 61-72, 1988.

[17] B. Potter, “Software security testing”, IEEE Security & Privacy Magazine, IEEE Computer Society, vol. 2, no. 5, pp. 81-85, 2004.

[18] M. Howard, “A Look Inside the Security Development Lifecycle at Microsoft”, MSDN Magazine, USA, 2005.

[19] M. Howard, “A Process of performing security code reviews”, IEEE Security & Privacy Magazine, vol. 4, no. 4, pp. 74-79, 2006.

[20] S. T. Siddiqui and M. U. Bokhari, “Selecting appropriate Requirements Management Tool for developing Secure Enterprises Software”, International Journal of Information Technology and Computer Science, vol. 6, no. 4, pp. 49-55, 2014.

[21] J. Whittaker, “Why secure applications are difficult to write”, IEEE Security & Privacy Magazine, IEEE Computer Society, vol. 1, no. 2, pp. 81-83, 2003.

[22] H. H. Thompson, “Why security testing is hard”, IEEE Security & Privacy, vol. 1, no. 4, pp. 83–86, 2003.

[23] M. I. Daud, “Secure software development model: A guide for secure software life cycle”, In Proceedings of the International MultiConference of Engineers and Computer Scientists (IMESC'10), Hong Kong, vol. 1, pp. 1-5, 2010.

[24] Website: https://www.infoq.com/articles/standish-chaos-2015. Accessed: August 28, 2017.

[25] The Chaos Report, the Standish Group International, Inc., [Online] 1994, Available online at: http://www.standishgroup.com/sample_research_files/chaos_report_ 1994.pdf . Accessed: August 7, 2016.

[26] A. Aurum and C. Wohlin, “Requirements Engineering: Setting the context”, In Engineering and Managing Software Requirements, A. Aurum and C. Wohlin (Eds.) Springer-Verlag, Berlin, Germany, pp. 1-15, 2005.

[27] S. Rosenberg, “Standish’s CHAOS Report and the software crisis”, [Online] 2006, Available online at: http://www.wordyard.com/2006/08/02/standishs-chaos-report-and-the-software-crisis. Accessed: August 7, 2014.

[28] [Online] The Chaos Report – 1995. Parthenon Consultancy Ltd, Available online at: http://www.parthenon.uk.com/project-failure-chaos.htm. Accessed: August 9, 2017.