# Position on Metrics for Security in Requirements Engineering

Mahwish Kundi
Department of Computer Science
University of Leicester
Leicester, UK
Email: kk262@le.ac.ukl

Ruzanna Chitchyan
Department of Computer Science
University of Leicester
Leicester, UK
Email: rc256@le.ac.uk

***Abstract*—A number of well-established software quality metrics are in use in code testing. It is our position that for many code-testing metrics for security equivalent requirements level metrics should be defined. Such requirements-level security metrics should be used in evaluating the quality of software security early on, in order to ensure that the resultant software system possesses the required security characteristics and quality.**

## I. Introduction

Traditionally, security vulnerabilities are usually detected after implementation, when testing a software system as a whole. Security requirements are often stated explicitly and, sometimes the potential misuse/abuse possibilities of the system are studied during requirements engineering. During testing, the security-related features of the software are tested, along with general software testing, using such traditional (non-specific to security) metrics as [1,2,3,4], for instance:

- Effectiveness defined as the percentage of the Number of defects identified during testing (D) to Number of test cases executed (i.e., (D/TE)*100);
- Efficiency which establishes a link between the number of defects identified during testing (D) and number of defects identified by users after release of the application (DU) (i.e., D/(D+DU)*100);
- Defect Density defined as a percentage of identified defects (D) and size of software entity (SS) (i.e., (D/SS)*100), etc.

Such testing has, un-doubtfully, proven to deliver systems of good quality [12], and has even motivated the dedicated test-driven development approach [11]. Yet, we believe that quantifying the quality of the systems security after its implementation is leaving it too late. Since many critical decisions on what security characteristics a given software system will end up with are made during Requirements Engineering (RE), we maintain that it is essential to test a systems security in RE. Position[1]: As a number of well-established software quality metrics are in use in code testing, so equivalent requirements-level metrics need to be defined for (security)1 testing in RE. Such metrics should be used in evaluating the quality of software (security) early on, in order to ensure that the resultant software system possesses the required (security) characteristics and quality. A brief review of current work on security metrics in RE is discussed in the following section, with a few initial proposals for new RE-level security metrics outlined in section III.

[1]This position equally applies to the broader set of requirements. Yet, here we are interested in security requirements and will thereafter focus on these only.

## II. Security Metrics in RE

According to a recent review of security metrics [6], it is observed that so far there are only a few metrics applied for assessing the security of software systems in RE. Yet, we already have some initial work in this area. Below we review some currently used metrics and their shortcomings: Correctness, effectiveness and efficiency of security are claimed to be fundamental measures in [7]. Where correctness implies that a system, its components, interfaces and the processed data meet the security requirements; effectiveness implies that expectations for resiliency in the use environment are satisfied and system works as intended; and efficiency implies that security quality has been achieved, meeting the resource, time and cost constraints. In [7] these metrics are simply defined and further work on actual metric collection and use is required. In [8] a framework is proposed to measure security risks throughout the entire software development life cycle (SDLC) with the help of Goal-Question-Metric approach. This paper defines 4-metrics for RE:

- Total number of security requirements is used as base line for defining RE measurement.
- Ratio of security requirements measures the ratio of the security requirements in a system (SR) to the set of all requirements of the system (R) (i.e., SR/R).
- Number of omitted security requirements is used to measure the omitted security requirements. Such security requirements are taken from the standard fixed in advance, for instance, the Common Criteria for Information Technology Security Evaluation (ISO/IEC15408).
- Ratio of the number of omitted security requirements defines the proportion of the number of security requirements that have not been considered (Nosr) to total number of security requirement (Nsr) (i.e., Nosr / (Nosr + Nsr) )

978-1-4799-6333-1/14/$31.00 © 2014 IEEE

RET 2014, Karlskrona, Sweden

Thus, though [8] proposes some baseline for further measurement in RE, it does not account for the issues of security correctness, effectiveness, coverage and efficiency, which we (in agreement with [7]) consider fundamental measurement objectives. In [9] a measurement approach to security requirements engineering is proposed which is aligned with the Security Quality RE (SQUARE) approach. This paper discusses security drivers and considerations. While [9] gives a theoretical approach of measuring security aligned to SQUARE, it also points out that SQUARE may not go far enough in identifying operational security requirements. There is a need for indicators that support quantified level of measurement. In [2] the Goal-Question-Metric approach is used to define and evaluate security measures. Here a step-by-step approach is taken: firstly SQUARE [9] is adopted for security requirements identification; then goals and respective metrics are set using Goal-Question-Metric approach. This work uses ISO/IEC 17799:2005 standard for information security as a baseline. In [10] Security Measurement Model (SMM) is introduced for evaluating the Degree of Security (DS). This model evaluates overall security of the target service through measuring the security in Security Requirements Model (SRM) of the service. This work measures degree of security in security requirements based on mitigation techniques. There is no clear view on general use of these metrics. The overall measurement is dependent on severity of security flaws while the rest of security issues, for example, number of security threats, how effectively threats have been removed etc. are not considered.

## III. Metrics for Quantification of Security in RE

As noted before, there are a number of mature metrics currently used for quantifying software quality in code testing, but not in requirements. It has also been observed that a use case and a test case have a close connection [13,14]. Measuring security at the requirement stage, focusing on misuse case model could mitigate security vulnerabilities before they reach the finalised product [15]. With this in mind, we could consider the relationship between use/misuse cases in requirements as parallel to test case/defects in code testing. Defects are the flaws related to a specific test case, whereas misuse cases are security flaws related to a use case. A *use case* (UC) is defined as a desired interaction between a user and the system to achieve a particular goal. A *misuse case* (MUC) is an interaction which must not be supported (i.e., a function which must not be performed) by the system. Intuitively, if a given use case brings up a larger number of security-related misuse cases, it is also likely to require more attention as part of systems security engineering. A few initial metrics for qualification of the security quality in RE are outlined below:

- RE Coverage is defined as the ratio of covered (i.e., tested) misuse cases to the total number of security-related use cases. This metric could help to set the threshold for when to stop evaluating misuse cases. Since each use case will have a number of misuse cases, some use cases (if considered non-critical) could be left uncovered in misuse analysis. We should also clarify that this metric is calculated only for the use cases relevant to security. (E.g., a system may have 500 use cases but only 100 of them could be relevant to security; coverage will then relate to these 100 use cases only). There is also a need for a clear process for identification of the relevant to security use cases this will be addressed in our immediate future work.
- RE Correctness is defined as the ratio of resolved misuse cases (MUCr) to the total number of identified misuse cases (MUC) (i.e., MUCr/MUC * 100). We should note that the resolution itself could be specified as a metric, for instance, as a threshold for acceptable risk level. Alternatively the misuse case can be considered resolved if one or more countermeasures are set for it. This metrics indicates the quality of security requirements. As resolution of a misuse case will necessitate definition of new requirements and constraints, it may be that not all identified MUC could be resolved, with some (deemed non-critical) left unresolved due to do budget and time considerations.
- RE Efficiency is defined as the ratio between number of misuse cases identified in RE (MUC) to the sum of these same misuse cases and those detected by the user (MUCu) (i.e., MUC/(MUC +MUCu)). This metric indicates how efficient the security analysis was in the RE phase. This metrics is particularly relevant when used with agile methodology where users have active involvement in RE.

Other metrics, such as density of misuse cases per use case, or ratio of time spent on misuse case analysis per use case and/or per security threat, etc. can also be defined. Finally, it is worth noting that security overall is a soft goal (i.e., comprises a number of related non-functional requirements). Where we propose functionality-based use/misuse cases for measuring security, we look at how well such relevant functionality is counter-measured against misuse, thus focusing on the security quality of the use/misuse case, not their function per-se.

## References


[1] J. A. Wang, H. Wang, M. Guo, M. Xia, Security Metrics for Software Systems 47th Southeast Conference, ACM, 2009.

[2] S. Islam and P. Falcarin, Measuring Security Requirements for Software Security, CIS 10th International Conference, IEEE, 2011.

[3] J. Dolecek, Metrics and Indicators on SoftwareTesting Process Effectiveness based on Requirements Coverage and Traceability Information, Master Thesis, Faculty of informatics Masaryk University, 2012 .

[4] J. Allen, Measuring Software Security, CERT Research Annual Report, Software Engineering Institute, Carnegie Mellon University, 2009.

[5] S. Jain and M. Ingle, A Review of Security Metrics in Software Development Process, IJCSIT, Vol. 2 (6), 2011.

[6] R. Savola, A security metrics taxonomization model for software intensive systems, JIPS, Vol.5 (4), 2009.

[7] A. Abdi, Using Security Metrics in Software Quality Assurance Process, 6'th International Symposium on Telecommunications, 2012.

[8] K. Sultan, Catalog of Metrics for Assessing Security Risks of Software throughout the Software Development Life Cycle ISA conference, IEEE, 2008.

[9] N.R. Mead, Measuring The Software Security Requirements Engineering Process (COMPSACW) 36th Annual conference workshop, IEEE, 2012.

[10] D. Mougouei, W. Rahman, M. Almasi, Measuring Security of Web Services in Requirement Engineering Phase IJCSDF, vol.1 (2), 2012.

[11] A. Causevic, S. Punnekkat, D. Sundmark,"Quality of Testing in Test Driven Development, QUATIC 8th International Conference, IEEE 2012.
[12] L. Madeyski, L. Szala, The Impact of Test-Driven Development on Software Development Productivity An Empirical Study, Software Process Improvement, Springer Berlin Heidelberg, 2007.
[13] C. Nebut, F. Fleurey, Y. L. Traon, J. Jezequel, Automatic Test Generation: A Use Case Driven Approach IEEE Transactions On Software Engineering Vol. 32(3), 2006.
[14] J. J. Gutierrez, M.J. Escalona, M. Mejias; J. Torres, A.H. Centeno, A Case Study for Generating Test Cases from Use Cases RCIS Second International Conference IEEE, 2008.
[15] A. A. Abdulrazeg, N. M. Norwawi, N.M, N. Basir, Security Metrics To Improve Misuse Case Model (CyberSec) International Conference IEEE, pp. 94-99, 2012.