International Journal of Cyber-Security and Digital Forensics (IJCSDF) 1(3): 211-220
The Society of Digital Information and Wireless Communications (SDIWC) 2012 (ISSN: 2305-0012)

Security Measurement Based On GQM To Improve Application Security During Requirements Stage

Ala A. Abdulrazeg[1]., Norita Md Norwawi[2]., Nurlida Basir[3]
Faculty of Science and Technology
Universiti Sains Islam Malaysia
Nilai, Malaysia
ala.alaker@yahoo.com[1], norita@usim.edu.my[2], nurlida@usim.edu.my[3]

ABSTRACT

Developing secure web applications that can withstand malicious attacks requires a careful injection of security considerations into early stages of development lifecycle. Assessing security at the requirement analysis stage of the application development life cycle may help in mitigating security defects before they spread their wings into the latter stages of the development life cycle and into the final version of product. In this paper, we present a security metrics model based on the Goal Question Metric (GQM) approach, focusing on the design of the misuse case model. Misuse case is a technique to identify threats and integrate security requirements during the requirement analysis stage. The security metrics model helps in discovering and evaluating the misuse case models by ensuring a defect-free model. Here, the security metrics are based on the OWASP top 10-2010, in addition to misuse case modeling antipattern.

KEYWORDS

Measurement; Security Metrics; Misuse cases; Security Requirements; Software Security.

1 INTRODUCTION

Web applications are employed in a wide variety of contexts to support many daily social activities. Unfortunately, the tremendous rise in online applications has been accompanied by a proportional rise in the number and type of attacks against them. Web applications are continuously reported to be vulnerable to attacks and compromises. According to a recent analysis conducted by Symantec Inc [1], vulnerabilities and security breaches on enterprises are increasing, with web application attacks continuing to be a favoured attack vector. Furthermore, a report by WhiteHat security has found that 8 out of 10 web applications are vulnerable [2]. These reports indicate that even present-day web applications are not free from vulnerabilities. In security engineering, vulnerabilities result from defects or weaknesses that are inadvertently introduced at the design and implementation stages of the development life cycle that can be exploited by attackers to harm the application and its asset [3]. Therefore, security needs to be considered and measured from the early stage of the development life cycle.

Mellado et al. [4] believe in the particular importance of security requirements engineering, which provide techniques and methods to handle security at the early stage of the software development lifecycle. A survey to identify and describe concrete techniques for eliciting security requirements showed that a misuse case is often considered an

important part of the requirement stage [5]. Misuse cases represent security threats that the attacker might interact with to breach security and cause harm to the system. Misuse cases are created by extending the use case model to provide a systematic way for identifying system functions, possible threats, and required countermeasures in one consistent view. The misuse cases model must be accurately modelled, because if security defects and vulnerabilities are discovered late in the development, the cost of fixing them escalates significantly as shown in table 1 [30].

Table1. Cost of fixing defects [30]

Defect Introduction Point	Defect Detected During				
	Requirements	Architecture	Constructior	Testing	After-Release
Requirements	1	3	5-10	10	10-100
Architecture	-	1	10	15	25-100
Construction	-	-	1	10	10-25

A study on security improvement program suggested that measurement and metrics must be included earlier in the development processes [8]. Measuring security at the requirement stage, focusing on misuse case model could mitigate security defects before they reach the finalised product. This paper proposes a new set of security metrics model that quantifies security at an early stage of web application development life cycle, namely the requirement stage. The security metrics are defined using the Goal, Question, Metrics approach. The proposed metrics model is misuse case-centric to ensure that the developed misuse case models are defect-free, and mitigate most well-known web application security risks. The security metrics model is defined by adopting the antipatterns proposed by [9] to ensure the modelled misuse cases are defect-free. The model is based on the prominent top 10-2010 web application security risks OWASP [10] to ensure the security use cases thoroughly address these risks.

The rest of the paper is organized as follows: Section 2 presents the background of the work which discusses the importance of security metrics and then presents the concept of the misuse case model. Section 3 presents the proposed security metrics model. In section 4, related work has been discussed. Finally, section 5 suggests future work and explains the conclusions.

2 BACKGROUND

2.1 Why Security Metrics

Metrics are defined as standards of measurement. Measurement is a process of quantifying the attributes of software to describe them according to clearly defined rules [11]. Chew et al. [12] defined measurements as the process of data collection, analysis, and reporting. The results of data collection are called ‘measures’. Lord Kelvin is known to have said, “If you cannot measure it, you cannot improve it. When you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meagre and unsatisfactory kind”[13]. The analysis and interpretation of appropriate measures helps diagnose problem and identify solutions during the development of software, which assists in reducing defects, rework, and cycle time [7].

According to Wang et al. [6] we cannot improve security if we cannot measure it. Security metrics are considered effective tools that allow information security experts to characterize and evaluate the effectiveness of security and the levels of systems, products, and processes in order to address security issues and facilitate

improvements [14]. Security metrics are used for decision support and these decisions are actually risk management decisions aimed at mitigating and cancelling security risks. Defining metrics based on goals has proven successful in guaranteeing relevant measurements, as it gives purpose to the metrics [15].

The Goal Question Metric approach is a goal-oriented approach which provides a framework for metrics development [15]. The GQM approach was originally developed by Basili and Weiss [16], and expanded by Rombach [17]. Basili [18] stated that the Goal Question Metric approach represents a systematic approach for integrating goals with models of the software processes, products and quality perspectives of interest, based upon the specific needs of the project and the organization. An example of GQM is illustrated in figure 1 [29].

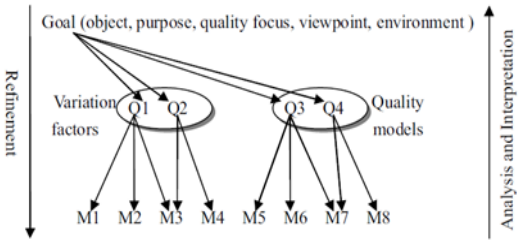

Figure 1. The Goal Question Metrics approach

As illustrated in figure 1, the goal Question Metric approach focuses on defining measurable goals (conceptual level) for products, processes, and resources with respect to quality issue perspectives of interest. Then, these goals are refined into questions (operational level) to characterize the way the assessment/achievement of these goals is going to be performed. Once the goals are refined into a list of questions, metrics are identified (Quantitative level) to provide a quantitative answer/information to each question in a satisfactory way [17].

2.2 Misuse Case Modelling

Ensuring the set of security requirements obtained is complete and consistent is a very important task because the right set of security requirements will lead to the development of secure software, whereas the wrong requirements can lead to a never-ending cycle of security failures [19]. Misuse case is a useful technique for eliciting and modelling functional security requirements and threats at the requirement stage.

Use case diagrams have proven effective during requirements engineering for selecting functional requirements, but offer limited support for selecting security requirements [20]. McDermott and Madison [21] used the term ‘abuse cases’ to express threats and countermeasures using the standard use case notation. In their approach, the authors kept the abuse case in separate models. Later, Sindre and Opdahl [22] extended the positive use case diagrams by adding negative use cases (misuses cases) to model undesirable behaviour in the system and misuser to model the attacker. Extending the use case model with misuse cases provides the ability to regard system functions and possible attacks with one consistent view, which assists in describing security threat scenarios which would threaten the system assets, mitigating threats and thus improving security. The ordinary use case relationships such as association, generalization, ‘include’ and ‘extend’ may also be used in modelling of misuse cases. Sindre & Opdahl [20] have refined the relationships in misuse case modelling by adopting threaten and mitigate relationships as suggested by [23]. These two types of relationships illustrate that a misuse case may *threaten* a use case, while a security use case might *mitigate* a misuse case.

A security use case represents software security requirements needed to protect system assets from security threats. The idea of security use cases as a way of representing countermeasures is presented by [24] and was adopted by [20].

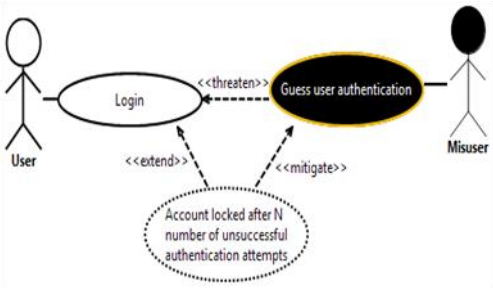

Figure 2. Misuse case diagram example

Figure 2 illustrates an example of a misuse case diagram. In this figure *Account locked after N number of unsuccessful authentication attempts* is a security use case added to protect against the threat *Guess user authentication* identified as a potential misuse case that threatens the *login* function.

3 SECURITY METRICS TO IMPROVE MISUSE CASE MODEL

In this work, we develop a security metrics model to be applied at the requirements stage. The proposed security metrics model is misuse case-centric and aims to discover and secure security vulnerabilities and modeling defects. It is significant to eliminate modelling defects from the misuse case model and improve security use cases before those defects and weaknesses find their way into the latter stages of the development life cycle.

The GQM approach is used for a structured and derivation of the security metrics. The proposed security metrics model is composed of two main goals. The first goal is to improve the quality of the developed misuse case models by ensuring the models are defect-free and do not contain any incorrect and misleading information. In order to achieve this goal, security metrics are developed based on the antipatterns specified by [9]. The antipatterns are the poor modelling decisions which result in low quality misuse case models that can lead to defects and harmful consequences in the latter stages of development life cycle [9]. The metrics have been scaled so as to fit within the range 0 to 1 with lower values considered a satisfactory rating for the measurement.

Goal 1 To improve the modeling quality of misuse case models by identifying modeling defects.

Question 1.1 Do the misuse cases correctly represent the application vulnerabilities and are they consistent with application security use cases?

> **Metrics 1.1.1** The ratio of the number of misuse cases that do not threaten the application to the total number of misuse cases.

Consider a set of misuse cases in a model as $MC = \{mc_1, \ldots, mc_n\}$ and the non-threatening misuse cases as $NMC = \{nmc_1, \ldots, nmc_n\}$ such that $NMC \subseteq MC$. The metric is expressed as follows, where RN_{MC} stands for the ratio of misuse cases that do not threaten the application.

$$RN_{MC} = \frac{NMC}{MC} \qquad (1)$$

> **Metrics 1.1.2** The ratio of the number of unmitigated misuse cases that threaten the application to the total number of misuse cases.

Consider a set of misuse cases in a model as $MC = \{mc_1, \ldots, mc_n\}$ and the unmitigated misuse cases as $UMC = \{umc_1, \ldots, umc_n\}$ such that $UMC \subseteq MC$. The metric is expressed as follows, where RU_{MC} stands for the ratio of the number of unmitigated misuse cases.

$$RU_{MC} = \frac{UMC}{MC} \qquad (2)$$

Question 1.2 Are the functional decompositions between misuse cases correctly handled?

Metrics 1.2.1 The ratio of inclusion misuse cases included once to the total number of inclusion misuse cases.

Consider a set of inclusion misuse cases as $IMC = \{imc_1, \ldots, imc_n\}$ and the inclusion misuse cases included once as $OIM = \{oim_1, \ldots, oim_n\}$ such that $OIM \subseteq IMC$. The metric is expressed as follows, where RO_{IMC} stands for the ratio of inclusion misuse cases included once.

$$RO_{IMC} = \frac{OIM}{IMC} \qquad (3)$$

Metrics 1.2.2 The ration of extension misuse cases extended once to the total number of extension misuse cases.

Consider a set of extension misuse cases as $EMC = \{emc_1, \ldots, emc_n\}$ and the extension misuse cases extended once as $OEM = (oem_1, \ldots, oem_n)$ such that $OEM \subseteq EMC$. The metric is defined as follows, where RM_{EMC} stands for the ratio of extension misuse cases extended once.

$$RM_{EMC} = 1 - \left(\frac{OEM}{EMC}\right) \qquad (4)$$

Metrics 1.2.3 The ratio of misuse cases used as pre/post conditions of other misuse cases to the total number of misuse cases.

Consider a set of misuse cases as $MC = \{mc_1, \ldots, mc_n\}$ and the misuse cases used as pre/post conditions as $PMC = \{pmc_1, \ldots, pmc_n\}$ such that $PMC \subseteq MC$. The metric is expressed as follows, where RP_{MC} stands for the ratio of misuse cases used as pre/post conditions.

$$RP_{MC} = \frac{PMC}{MC} \qquad (5)$$

Question 1.3 Are the misusers presented and handled correctly in the misuse case model?

Metrics 1.3.1 The ratio of the number of the base misuse cases associated to one misuser to the total number of base misuse cases.

Consider a set of base misuse cases in a model as $MC = \{mc_1, \ldots, mc_n\}$ and the base misuse cases associated to one misuser as $OMM = \{omm_1, \ldots, omm_n\}$ such that $OMM \subseteq MC$. The metric is expressed as follows, where RM_{MC} stands for the ratio of the number of the base misuse cases associated to one misuser.

$$RM_{MC} = 1 - \left(\frac{OMM}{MC}\right) \qquad (6)$$

The second goal of the security metrics is to discover omitted security use cases that mitigate known-security vulnerabilities to ensure that the developed misuse cases cover these vulnerabilities. To achieve this goal security metrics based on web application security risks OWASP top 10-2010 [10] were developed. In this work, three security risks were analyzed; SQL injection, Cross Site Scripting, and Broken Authentication and Session Management.

Goal 2: To ensure that the elicited security use cases cover the well-known security vulnerabilities.

Question 2.1 What is the number of misuse cases found?

Metric 2.1.1 The total number of identified misuse cases [MUC_{Total}].

Question 2.2 What is the number of elicited security use cases?

Metric 2.2.1 The total number of elicited security use cases [SUC_{Total}].

Question 2.3 Are the security requirements which have been defined sufficient to mitigate well-known security vulnerabilities?

Metric 2.3.1 The number of excluded security requirements that ensure input/output handling [Xr_1].

Is a specific encoding scheme defined for all inputs?	
Is a process of canonicalization applied to all inputs?	
Is an appropriate validation defined and applied to all inputs, in terms of type, length, format/syntax and range?	
Is a whitelist Filtering approach is applied to all inputs?	
Are all the validations performed on the client and server side?	
Is all unsuccessful input handling rejected with an error message?	
Is all unsuccessful input handling logged?	
Is output data to the client filtered and encoded?	
Is output encoding performed on server side?	

Metric 2.3.2 The total number of excluded security requirements that ensure Authentication & Authorization handlin [Xr_2].

Is a complex password policies applied in order to choose proper passwords?	
Is the minimum and maximum length of password defined?	
Is the account automatically locked for the specified period when a specified number of consecutive unsuccessful authentication attempts exceeded?	
Is authentication error messages not verbose and do not contain sensitive information?	
Is the option that remembers the authentication credentials such as "Keep me signed in" avoided?	
Is user allowed to change his/her password?	
Is user allowed to create his/her own secret questions and answers for the option of password recovery?	
Is CAPTCHA (Completely Automated Turing Test To Tell Computers and Humans Apart) applied?	
Is all authentication decision performed on the server side?	
Is all authentication actions (e.g, Login, logout, password change) logged?	
Is re-authentication required when performing critical operations?	
Is user forced to change Password after a specific period of time (expiration periods)?	
Is user credentials rejected without even validation when the account is locked?	
Is secure data transmission protocol applied to secure credentials transfer between client and server.	

Metric 2.3.3 The total number of excluded security requirements that ensure session handling [Xr_3].

Is session identifier created on server side?	
Is new session identifier assigned to user on authentication?	
Is session identifier changed on re-authentication?	
Is logout option provided for all operations that require authentication?	
Is session identifier cancelled when authenticated user logs out?	
Is session identifier killed after a period of time without any actions?	
Is user's authenticated session identifier protected via secure data transmission protocol?	

Metric 2.3.4	The total Number of excluded security requirements that ensure Error & Logging handling [Xr_4].

Is application has log file?	
Is log control handled on server?	
Is the application does not output error messages that contain sensitive data?	
Is all server failure and errors handled in server and NOT deliver to user?	

These metrics are implemented by comparing the elicited security requirements of the application during the requirement stage to the stated security requirements. These metrics assess the threat of possible attacks on the system. If a security requirement has been excluded then a value of 1 will be given, and a value of 0 if it has been considered.

Metric 2.3.5 The total number of excluded security requirements that put the system at risk of possible attacks.

$$ExR_{SUC} = \sum_{i=1}^{n} Xr_i \qquad (7)$$

ExR_{SUC} stands for the summation of the excluded security requirements, and Xr_i represents the excluded security requirements that put the system at risk, where i $\in \{1, 2, ..n\}$.

Question 2.4 How vulnerable is the application based on the stated security requirements?

Metric 2.4.1 The ratio of the number of included security requirements to the total number of stated security requirements.

$$RV_{SUC} = 1 - \left(\frac{SsR - ExR_{SUC}}{SsR} \right) \qquad (8)$$

SsR stands for the total number of the stated security requirements. The difference between SsR and ExR_{SUC} indicates the included security requirements. RV_{SUC} stands for the ratio of the number of included security requirements. The value of the metrics ranges from 0 to 1. If RV_{SUC} converges to 0, that indicates many stated security requirements have been considered in the misuse case model. The lower ratio is the satisfactory rating for the measurement. The security metrics model is illustrated graphically in figure 3.

4 RELATED WORK

A number of related works have already been done that introduce security metrics, or mentioned how and where to situate theses metrics in the development life cycle of a system. Nichols and Peterson [25] introduced a metrics model based on OWASP top-10 vulnerabilities and organized according to the application's life cycle. The authors suggested that if the organization seeks to improve the overall application security, they must focus on security of the web application itself. The authors also suggested that web application developers need to be concerned about the vulnerabilities that may exist in the application. In this paper, the authors stated that design-time metrics are essential to the application development because of their ability to identify and characterize weaknesses early in the application's life cycle. Mell et al [26] reported the Common Vulnerability Scoring System (CVSS) provides an open standardized tool to measure the severity and risk of a vulnerability discovered in a given system. CVSS assists in prioritizing these vulnerabilities to remediate those that pose the greatest risk. Chowdhury et al [27] defined a number of security metrics that assess how securely a system's source code is structured. The proposed metrics can be applied to evaluate the robustness, secure

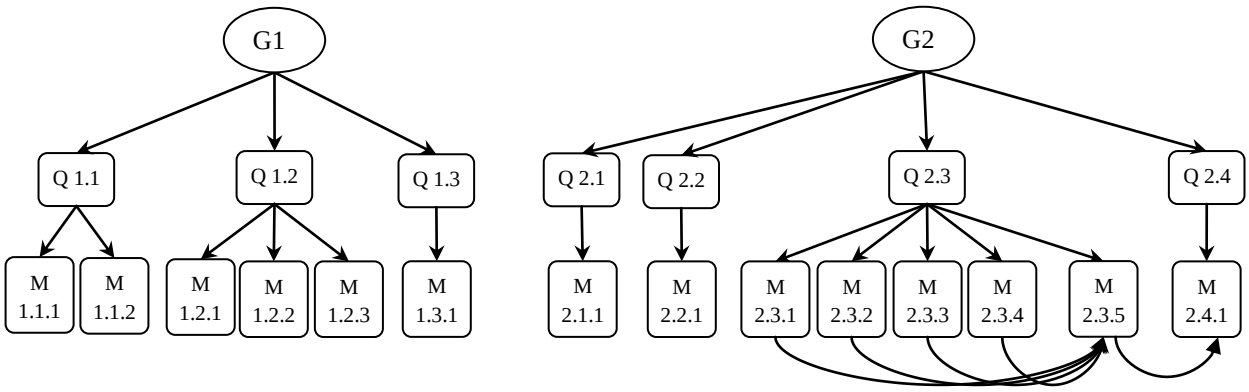

Figure 3. Graphical representation of the security metrics model based on GQM

information flow and secure control flow in code structures. Wang et al [6] described an approach to define security metrics based on vulnerabilities included in software systems and their impact on software quality. The proposed security metrics measure the severity level and the risk of a representative weakness of software that causes most of the vulnerabilities to be exploited by the attackers, taking into consideration the time of occurrences of vulnerabilities at the software product level. Alshammari, et al [28] proposed a set of security metrics to measure information flow of object-oriented designs based on the analysis of quality design properties presented in the Quality Model for Object-Oriented Design. These properties include: composition, coupling, extensibility, inheritance, and design size. The author studied each property and its relevance to designing secure software to define the security metrics.

5 CONCLUSIONS

In today's world, security is an important aspect of web application. A prudent approach for developing security web applications is to integrate security from the early stages of development, specifically at the requirements stage. This paper provides a security metrics model to examine the misuse case diagram to ensure it is defect-free, and covers and mitigates known-security risks and vulnerabilities, so as to develop a secure system. The proposed security metrics give an indication of where the security defects might occur. Future works may consider conducting experiments to evaluate and demonstrate the usefulness and effectiveness of the proposed security metrics for the system development. The effectiveness of the approach could be validated by evaluating the resulting misuse case diagram to fix defects in the original model and threats that are added to the model that could jeopardize the application.

6 ACKNOWLEDGMENT

The first author gratefully acknowledges the Ministry of Higher Education in Libya for sponsoring his PhD studies. The authors would like to acknowledge the support of the Faculty of Science and Technology at Universiti Sains Islam Malaysia for funding this work through the project No.PPP/FST-1-15711.

7 REFERENCES

[1] Symantec Inc. "Symantec Global Internet Security Threat Report Trends for 2009". Symantec Global Internet Security Threat Report. Volume XV,p.7 (2010).

[2] J.Grossman,"10 important facts about website security and how they impact your enterprise". WhiteHat Security .p.3 (2011).

[3] G. Elahi, E. Yu, and N. Zannone, "A vulnerability-centric requirements engineering framework: analyzing security at countermeasures, and requirements based on vulnerabilities", Requirements Engineering, '. pp. 41-62 (2010).

[4] D. Mellado, C. Blanco, L. S´anchez and E. FernadezMedina," A systematic review of security requirements engineering". Computer Standards & Interfaces. pp. 153-165 (2010).

[5] I. Tondel, M. Jaatun and H. Meland, "Security requirements for the rest of us: A survey". IEEE Software, Vol.25. pp.20–27 (2008).

[6] J. Wang, H. Wang, M. Guo and M. Xia, "Security metrics for software systems" .In Proceedings of the 47th Annual Southeast Regional Conference, South Carolina (2009).

[7] J. McCurley, D. Zubrow and C. Dekkers, "Measures and Measurement for Secure Software Development" Software Engineering Institute (2008).

[8] D. Taylor and G. McGraw," Adopting a software security improvement program". IEEE Security & Privacy. pp. 88-91 (2005).

[9] M. Elattar, "A framework for improving quality in misuse case models", Business Process Management Journal.pp.168 -196 (2012).

[10] J. Williams and D. Wichers, " OWASP top 10 – 2010" Technical report, The open web application security project (OWASP) (2010).

[11] C. Kaner, and W. Bond, " Software Engineering Metrics: What Do They Measure and How Do We Know". In: Proce 10th International Software Metrics Symposium, Chicago, USA. pp 1-12 (2004).

[12] E. Chew, S. Marianne, S. Kevin, B. Nadya, B. Anthony and R. Will, "Performance measurement guide for information security". Research Technical Report, NIST National Institute of Standards and Technology, Special Publication 800-55. July (2008).

[13] S. Bellovin, "On the Brittleness of Software and the Infeasibility of Security Metrics". IEEE Security & Privacy. pp. 96 (2006).

[14] A. Wang, "Information security models and metrics". In: Procs of the 43rd annual Southeast regional conference – VOL: 2, NY, USA. pp.178-184 (2005).

[15] B. Patrik and J. Per, "A Goal Question Metric Based Approach for Efficient Measurement Framework Definition". In: Proc of the 2006 ACM/IEEE international symposium on Empirical software engineering. Rio de Janeiro. pp 316 – 325 (2006).

[16] V. Basili and D. Weiss, "A Methodology for Collecting Valid Software Engine-ering Data", IEEE Tram. Software Engineering. Vol.10, No.6. pp.728-738 (1984).

[17] V. Basili, G. Caldiera and D. Rombac, "Goal Question Metric Paradigm", Encyclopedia of Software Engineering, Vol. , pp. 528-532 (1994).

[18] V. Basili, "Software Modeling and Measurement: The Goal Question Metric Paradigm," Computer Science Technical Report Series , University of Maryland, College Park, MD (1992).

[19] K. Beznosov and B. Chess "Security for the rest of us: An industry perspective on the secure-software challenge". IEEE Software, Vol. 25.pp.10–12 (2008).

[20] G. Sindre and A. Opdahl,"Eliciting Security Requirements with Misuse Cases", Requirements Engineering Journal.pp. 34-44 (2005).

[21] J. McDermott and J. Madison. Using Abuse Case Models for Security Requirements Analysis (1999).

[22] G. Sindre and L. Opdahl, "Eliciting security requirements by misuse cases". In: Proc of TOOLS Pacific 2000, Sydney, Australia (2000).

[23] I. Alexander, "Modelling the interplay of conflicting goals with use and misuse cases". In: Proc of the 8th international workshop on requirements engineering: foundation for software quality (REFSQ'02), Essen, Germany (2002).

[24] D. Firesmith, "Engineering security requirements", Journal of Object Technology. pp 53–68 (2003).

[25] E. Nichols and G. Peterson, "A metrics framework to drive application security improvement", The IEEE computer society. pp 88–91 (2007).

[26] P. Mell, K. Scarfone and S. Raomanosky, "A complete guide to the common vulnerability scoring system version 2.0"

in Forum of Incident Response and Security Teams (FIRST). pp 1-23 (2007).

[27] I. Chowdhury, B. Chan and M. Zulkernine, " Security Metrics for Source Code Structures" In: Proc of the Fourth International Workshop on Software Engineering for Secure Systems, Leipzig, Germany. pp 57-64 (2008).

[28] B. Alshammari, C. Fidge and D. Corney, "Security metrics for object-oriented designs". In: Proc 21st Australian of Software Engineering Conference, Brisbane, Australia. pp. 55–64 (2010).

[29] Xu. T. "Composite Measurement Pattern". In: proc of WiCOM '08. 4th International Conference. Dalian, China, pp.1-6 (2008).

[30] McConnell. S. "Code Complete". Microsoft Press (2004).