Journal of Theoretical and Applied Information Technology
10th June 2016. Vol.88. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

METHODOLOGICAL FRAMEWORK FOR ANALYSIS AND SYNTHESIS OF A SET OF SECURE SOFTWARE DEVELOPMENT CONTROLS

[1]ALEXANDER VLADIMIROVICH BARABANOV, [2]ALEXEY SERGEEVICH MARKOV, [2]VALENTIN LEONIDOVICH TSIRLOV

[1]NPO Echelon, Elektrozavodskaya str., 24, Moscow, 107023, Russia
[2]Bauman Moscow State Technical University, 2-nd Baumanskaya, 5, 1, Moscow, 105005, Russia

ABSTRACT

This article discusses the issues of standardizing commercial production of secure software products. It studies administrative and technical controls for minimizing the number of vulnerabilities during development and operational support of secure automated systems software. We classified standards and guidelines on secure software development. We analysed applicability of the available methodological approaches to secure software development during evaluation of conformance to the information security requirements, including during certification of the software products. The article proves feasibility of harmonizing the developed regulatory requirements and practical measures with the methods stipulated in ISO 15408 and ISO 12207 international standards. We introduced the notion of secure software and developed a basic set of requirements that allows, among other things, evaluating the conformance of the software development processes to secure software requirements. We prove that the set of requirements shall rest, first of all, on the accepted security policies and up-to-date threats. Sample requirements under development are provided. We developed an original conceptual model for analysis and synthesis of a set of controls for secure software development that rests on a set of generated requirements. The article shows that the conceptual model gives the software developers an opportunity to make a science-based choice of software development controls. We developed general methods for selecting a set of secure software development controls. We provide indirect proof of efficiency of the suggested approach. It should be noted that the suggested approach was used as a basis for the national standard on development and production of secure software.

Keywords: *Software Security, Secure Software Development Lifecycle, Certification, Information Security, Vulnerabilities*

1. INTRODUCTION

Information security (IS) of computer systems is currently exposed to threats that are mainly related to known unclosed vulnerabilities and unknown vulnerabilities. In addition, our country has virtually no regulations on secure software products development, which would be aimed at minimizing the number of IS vulnerabilities that are generated at various stages of software lifecycle. Therefore, there is an unsolved issue of special inspections (licensing) of companies, and inspection of production process (conformance evaluation) of serial software products as to information security requirements [1, 2].

This defines the target of this study, which consists in analysis and synthesis of a set of secure software development controls. This article explains how this target can be achieved.

2. ANALYSIS OF SECURE SOFTWARE DEVELOPMENT STANDARDS

The software development industry currently has a set of corporate, industry and international standards that describe controls and mechanisms aimed at secure software development, for instance [3-12]:

- ISO 15408;
- ISO 27034-1;
- ISO TR 24772;

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Microsoft Security Development Life Cycle;
- Cisco Security Development Life Cycle;
- OpenSAMM;
- OWASP CLASP.

The results of comparative analysis of the above standards are specified in Table 1.

Table 1. Secure Development Standards

Description	Standard						
	ISO 15408	Microsoft SDL	Cisco SDL	Open SAMM	OWASP CLASP	ISO TR 24772	ISO 27034-1
Controls related to employee training in secure software development by the developer (company)	-	+	+	+	-	-	-
Controls related to ensuring security of the software development infrastructure	+	-	-	-	-	-	-
Controls related to management of configuration of the software under development	+	-	-	-	-	-	-
Controls related to modelling information security threats, which source is found in software	+	+	+	+	+	-	-
Controls related to defining requirements for secure software development, which are applicable to software being developed	+	+	+	+	+	-	-
Controls related to using a standard for software source code execution (rules and guidelines aimed at minimizing the quantity of potentially vulnerable designs in the software source code)	-	+	+	+	+	+	-
Controls related to performing static analysis of the software source code	-	+	+	+	+	-	-
Controls related to dynamic analysis of the software dynamic code	-	+	+	+	+	-	-
Controls related to expert evaluation of software source code in manual mode	-	+	+	+	+	-	-
Controls related to analysis of vulnerabilities	-	+	+	+	+	-	-
Controls related to ensuring safe delivery of software to the user	+	+	+	+	-	-	-
Controls related to eliminating software vulnerabilities identified during the software operation	+	+	+	+	+	-	-
Possibility to use the document to assess the conformance of software to the requirements for secure software development	+	-	-	-	-	-	-
Methods for selecting subsets of secure software development controls	+	+	+	+	-	-	+
Correspondence with the software lifecycle processes defined in ISO 12207	-	-	-	-	-	-	+

The following conclusions were made based on the analysis results:

1. There are currently a number of rather detailed standards devoted to secure software development issues. Such standards, as a rule, pursue the following goals:
   - To create software with a minimum number of vulnerabilities possible and to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

generate an environment for quick elimination of software problems (software vulnerabilities) identified by the users;

- To implement software elements of security policy for the environment (location) of the software application (IS policy for an organization or country, where software is used).

2. The reviewed documents suggest using a set of secure software development controls at various stages of the software lifecycle to achieve the goal of creating software with the minimum number of vulnerabilities and creating an environment for elimination of the identified problems. The suggested list of secure software development controls is standard, as a rule, and contains controls, related to, for example, modelling threats, static and dynamic analysis, penetration testing [12-17].

3. The above documents do not describe a clear framework, which may be used to assess the conformance of the software development processes to the requirements to the secure software development, for instance, within the framework of the software certification. It should be noted that international standard ISO 15408 is currently widely used during the software certification as to information security requirements, but the use of provisions of this document alone during development and further assessment of compliance of software security is not sufficient, because:

- The standard applies only to software with security functions, in other words, the security target required by the standard may not be written for the software without implemented security functions, neither can the conformance assessment be performed accordingly;
- The list of controls for software development suggested in the third part of the standard is not comprehensive: particularly, there are no controls related to static or dynamic analysis, employee training etc.

3. BASIC SET OF REQUIREMENTS

It is a well-known fact that the current development of the certification system for information security tools as to information security requirements is related to assessment of Common Criteria (ISO 15408) [3, 4, 9, 15, 18]. That is why we made a decision to use ISO 15408 methods in this study for development of a conceptual model for analysis and synthesis of secure software development controls.

When defining the software security environment we describe the following security aspects of the environment, where the software is going to be used:

- Security assumptions that contain security aspects of the environment, where the software is going to be used;
- Information security threats that are related to the assets, which require protection by means of software or its environment;
- Security policies of the company, which identify, and, if necessary, explain the provisions of the company security policy or the rules the software should conform to.

The security goals reflect the stated intention to withstand all established IS threats, and to cover all security assumptions and the established information security policy of the company. The description of the software security requirements shall define the functional requirements (for instance, requirements to identification/ authentication or access control) and assurance requirements that the software and the process of its development should meet. Functional security requirements and assurance requirements to security should be selected from the catalogue of requirements, which can be found in the 2nd and 3rd parts of the standard. Lists of software security functions and software development controls should be compiled to meet the identified functional security requirements and assurance requirements. All identified sets (security environment aspects, goals, requirements, controls) should correlate (this information and the required explanations are provided in the security target, as a rule).

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

To adapt the procedure for generating and substantiating the list of software security functions and software development controls and to achieve the goal of this study we developed a basic set of requirements for secure software development. The following specifics were taken into account, when forming a basic set of requirements for secure software development:

1. To ensure practical application and effective fulfilment of requirements the set being developed shall be correlated with International Standard ISO 12207, which describes the processes of the software life cycle [13].
2. It is a well-known fact that it is more expensive to eliminate software vulnerabilities during later stages of software lifecycle (for instance, during software testing or support), therefore the set of requirements under development shall ensure implementation of the required measures at the earliest stages of the software development [8, 9].
3. To meet the target for development of framework for assessment of software conformance to the requirements being developed during generation of certain requirements we took into account the aspects related to:
    - Generating requirements to documented evidence of the requirement fulfilment;
    - Generating requirement to the actions of the assessor (for instance, test laboratory expert), that are met during assessment of the software compliance with the secure software requirements.
    - In the course of the study we decided to use the following parameters to describe a requirement for secure software development to account for the identified specifics:
    - Name of the requirement;
    - Unique requirement ID;
    - Reference to software lifecycle process, which is established by ISO 12207, during which the requirement may be fulfilled [22];
    - Goal related to secure software development that is achieved by the developer during fulfilment of this requirement;
    - Elements of the developer's actions, which describe the actions of the software developer aimed at meeting the requirement;
    - Elements of the content and provision of the documented evidence of the requirement fulfilment;
    - Elements of the assessors' actions, which describe the actions of the assessor aimed at independent verification of the requirement fulfilment by the software developer;
    - Explanatory notes.

In the course of the study we designed a set of $B = \{\beta_1, \beta_2, \ldots, \beta_{24}\}$ requirements for secure software development based on existing standards and methods of secure software development [10, 18-20]. Table 2 briefly describes the developed basic set of requirements for secure software development [21].

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

Table 2. Suggested Basic Set of Requirements for Secure Software Development

Brief description of the requirement
The software developer shall define the requirements to secure software development, which are imposed on the software being developed
The software developer shall model information security threats that originate in the software to identify potential threats to information security and substantiate software requirements
The software architecture (software logic structure, which identifies components, their interfaces and the concept of interaction between them) shall be updated with account for the need to neutralize potential threats to information security, which are identified at the stage of information security threats modelling, and performance of the requirements to secure software development, which are established during analysis of software requirement
The software developer shall identify each tool used to develop the software
The software developer shall create a program based on the software architecture, defined during the process of designing and detailed designing of the software architecture
The software developer shall create or select a standard, which contains a list of rules and recommendations, aimed at minimizing the number of potentially vulnerable structures in the source code, for the source code execution
The software developer shall perform static analysis of the source code to detect potentially vulnerable structures in the source code.
The software developer shall regularly perform expert evaluation of the source code to identify potentially vulnerable structures in the source code
The software developer shall perform functional testing of the software to determine whether the requirements, which have been identified during analysis of the software requirements, are met
The software developer shall perform a penetration testing of the program to identify software vulnerabilities
The software developer shall perform dynamic analysis of the software code to identify software vulnerabilities
The software developer shall use technical and administrative controls, required for identifying software modifications or any discrepancies between the original and the version, received by the user
The delivered software shall have operating instructions within the scope sufficient for correct setting up and safe use of the software
The software developer shall implement the procedure that allows for tracing and eliminating the detected software vulnerabilities
The software developer shall be able to offer the user a solution to the problem in the situation, when previously unknown software vulnerability is used for computer or network attack on the user's information system
The software developer shall implement the procedure that allows for unique marking of each software version
The software developer shall use the system for software configuration management, which allows for unique identification of the elements of configuration that are related to the software being developed
The system for software configuration management shall identify the elements of configuration, which are related to implementation of the secure software functions
The system for software configuration management shall provide tools for identifying all elements of the configuration, which are affected by the modification of this element of configuration
The software developer shall use technical and administrative controls that ensure protection from unsanctioned access to the elements of configuration
The software developer shall use technical and administrative controls that ensure data back-up and restoration of the configuration elements
The software developer shall use technical and administrative controls that ensure registration of all events related to the facts of changes in the configuration elements in the event log

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

	Brief description of the requirement
	The software developer shall provide regular training to the employees to increase their awareness in the area of secure software development
	The software developer shall regularly analyse the employee training programs to establish their feasibility, adequacy and efficiency in helping to achieve the goals of secure software development

Static analysis of the software security is the most effective (unfortunately, it is rather time consuming [15]), therefore in this article we found it appropriate to give an example of the developed requirement (β_7) (Table 3).

Table 3. Requirement for Statistic Analysis of the Software Source Code

Parameter	Value
Name	Source code static analysis
Requirement No	7 (β_7)
Software life cycle process	Construction and integration process
Goal	1. To identify and correct potentially vulnerable designs in the source code; 2. To generate source data to perform dynamic analysis and penetration testing as part of the software qualification testing.
Elements of the developer's actions	SW developer shall perform static analysis of the source code to identify potentially vulnerable designs in the source code. Static analysis of the source code shall be used for the components borrowed from the external software developers, if the source code is available for them. Based on the static analysis of the source code the program might require debugging. If such debugging is not required or impossible, the developer shall document the substantiation of this fact.
Elements of the content and provision of the documented evidence	Documented evidence of the source code static analysis shall provide: - Information about the frequency of the source code static analysis; - Name and identification attributes of the tools used for static analysis of the source code; - The list of detected potentially vulnerable designs in the source code (if any), description of actions aimed at their elimination, or substantiation of impossibility or lack of necessity in the program debugging.
Elements of the assessor's actions	1. The assessor shall study the submitted evidence and confirm that they satisfy the specified requirements. 2. The assessor shall make independent findings that the developer performs static analysis of the source code based on the results of the survey of the employees of the software developer's company, which are related to software development, analysis of the environment of the software development.
Remarks	The software developer or external organizations that are qualified to detect software vulnerabilities perform static analysis of the source code for the up-to-date version of the source code. Static analysis of the source code allows finding potentially vulnerable designs in the source code, which may result in software vulnerabilities, and verifying the conformance of the source code to the standard of the source code execution adopted in the company.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

The developed basic set of requirements for secure software development was used to create a conceptual model for analysis and synthesis of a set of secure software development controls, which can be found below.

4. THE RESULTS OF DEVELOPMENT OF A CONCEPTUAL MODEL

Let us introduce the following sets for description of the conceptual model for analysis and synthesis of measures for the secure software development. The following sets should be defined to describe the software development environment:

- $U = \{u_1, u_2, \dots u_u\}$ is a set of information security threats, which are relevant for the software development environment;
- $P = \{p_1, p_2, \dots, p_b\}$ is a set of identified provisions of the security policy, which the software development environment should comply with.

Information security threats are described informally using the following parameters:

- Description of the information security threat;
- Source of the information security threat;
- Implementation method of the information security threat;
- Vulnerabilities used;
- Type of information resources in the software development environment that are potentially subject to the information security threat;
- Violated security property of the information resources in the software development environment;
- Possible consequences of the information security threat implementation.

The provisions of the security policies are described informally (in the natural language). It should be noted that the traditional approach described in the Common Criteria includes the description of the secure environment using an additional set – the set of secure usage assumptions.

During the analysis and synthesis of the secure software development measures the software development environment is clearly identified and defined, therefore, the formation of the assumption list is redundant.

A set of secure software development goals $O = \{o_1, o_2, \dots o_c\}$ is formed to achieve identified security policies and to neutralize the identified information security threats.

Based on the established goals for secure software development a subset of requirements $R = \{r_1, r_2, \dots r_d\}$ is selected from the basic requirements for secure software development.

To fulfil identified requirements, a software developer generates and implements a set of measures for secure software development $C = \{c_1, c_2, \dots c_s\}$ in the software development environment.

The following mappings were introduced:

- $F_O: U \cup P \to O$ is a procedure for generating goals for secure software development;
- $F_R: B \cup O \to R$ is a procedure for selecting secure software development requirements;
- $F_C: R \to C$ is a procedure for synthesis of measures for secure software development.

The developed model is characterized by tuple $\langle B, U, P, O, R, F_O, F_R, F_C \rangle$.

It should be noted that during development of the model we assume that all requirements are non-functional, in other words, they are similar to assurance requirements as per ISO 15408-3 (Figure 1).

Journal of Theoretical and Applied Information Technology
10th June 2016. Vol.88. No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

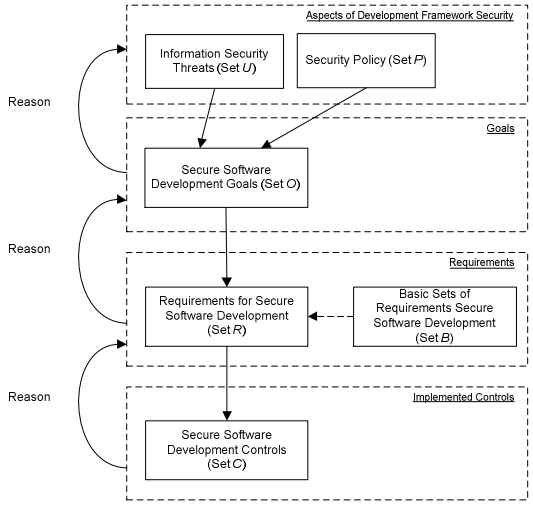

Figure 1. Generation Of A Set Of Controls.

In the course of the study, we developed general methods for analysis and synthesis of secure software development controls, which consist in implementation of the following stages and steps:

Stage 1. Identify and describe the aspects of secure environment for software development.

- Step 1. Form a set of information security threats $U = \{u_1, u_2, \dots u_a\}$, which are relevant for software development environment.
- Step 2. Form a set of guidelines of security policies $P = \{p_1, p_2, \dots, p_b\}$, to which the software development environment should conform. The sources for generating sets of the security policies can be represented by the requirements of laws, regulatory legal acts, industrial standards, a list of the user's requirements, and software use scenario.

Stage 2. Form and substantiate sets of goals for secure software development.

- Step 3 Form sets of goals for secure software development: $O = F_O(U, P)$.
- Step 4. Substantiate the completeness and sufficiency of the formed set of goals for secure software development. The substantiation of the goals for secure software development shall show that the set goals cover all identified security aspects (sets of information security threats and guidelines of security policies) of software development environment. This should be demonstrated as follows:

a. Substantiate completeness using cross references: goals related to development of secure software aimed at accounting for identified sets of information security threats and provisions of the security policies, should be represented as a table that should demonstrate that:
   - - Each goal covers at least one information security threat or provision of the security policy;
   - - Each information security threat or provision of the security policy is covered by at least one goal for development of secure software.

b. Substantiate sufficiency of the set of goals related to secure software development: it is necessary to demonstrate that the set of goals related to development of secure software is sufficient to account for all aspects of the software development. For this purpose, the Table of Software Development Environment Goals and Aspects Conformance shall be supplemented with the following informal explanation:
   - - It is necessary to demonstrate for each information security threat that the specified goals will allow neutralizing this information security threat;
   - - It is necessary to show for each provision of the security policy in which way the specified goals ensure its performance.

Stage 3. Select and substantiate the requirements for secure software development.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

- Step 5. For a set of requirements for secure software development on the basis of the basic requirements with account for the need to achieve the identified goals: $R = F_R(B, O)$.
- Step 6. Substantiate the completeness and sufficiency of the selected set of requirements for secure software development. The substantiation shall be made in the same way as the substantiation described in Step 4 using cross references and informal explanatory test.

Stage 4. Synthesize and substantiate measures for secure software development.

- Step 7. On the basis of the set of the identified requirements for secure software development and information about the specific software development environment, technologies and tools used during development, it is necessary to form sets of measures for development of secure software, which are going to be applied in the software development environment: $C = F_c(R)$.
- Step 8. Substantiate the completeness and sufficiency of the synthesized sets of measures for secure software development. The substantiation shall be made in the same way as the substantiation described in Step 4 using cross references and informal explanatory test.

5. TRANSITION TO MATHEMATICAL MODELS

The suggested conceptual model may also be used as the basis for developing mathematical models of secure development, for instance, multifactorial model:

$$P = \sum_{i=1}^{K} k_i p_i + \sum_{i \neq j}^{K} k_{ij} p_i p_j + \sum_{i=1}^{K} k_{ii} p_i^2 + \cdots$$

,

where: ki – coefficient of i- th control; $K \leq 24$ - number of controls.

It should be noted that we did not aim to receive the values for the above mathematical indicators, however, the efficiency of the suggested conceptual approach was confirmed by the results of comparison of the vulnerability level of the software products, developed in companies with recognised international certificates for IS management systems, and products, developed by other software developers. The analysis of statistics for 2008-2015 showed that the total number of vulnerabilities in the products of the first category is 5 times lower than the similar indicator in the products of the second category [23].

6. DISCUSSION

The drafted basic set of requirements for secure software development and the conceptual model for secure software development analysis/synthesis received approval during the discussions that involved the leading Russian experts in secure software development. They were also approved by the Technical Committee for Standardization when discussing the national standard GOST R “Software protection. Development of secure software. General requirements”. In the course of discussions, the experts underscored the following positive aspects of the developed basic set and the analysis/synthesis model:

1. Wide scope of application of the obtained results:
   - Both software developers and evaluators (when evaluating designer-implemented processes for meeting the requirements specified) can take advantage of the results offered in the work;
   - The basic set of requirements offered in the work applies to all software types, not only to those with security functions;
   - The set of requirements applies to software of any size and type, as distinct, for example, from the one proposed in the work [8], which can only apply to

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

operating systems unless drastically modified.

1. The designed conceptual model for secure software development analysis/synthesis helps, in a deterministic and well-founded manner, to identify a package of measures for secure software development based on threat analysis and the provisions of security policies.
2. Easy integration of the secure software development measures into the designer's current software development lifecycle processes by tracing the basic set of ISO 12207 lifecycle process requirements.

The following limitations of the developed basic set of requirements and the conceptual model were defined in the research data approval process:

1. The proposed requirements were not classified, for example, by the software scope, size and other characteristics that may potentially affect the range of measures applied in designing secure software.
2. The proposed basic set of requirements is static; no formal mechanism was offered for adding new requirements to the basic set.
3. The phase of generating the conceptual model for secure software development analysis/synthesis did not involve reviewing the issues of developing the basic information security threat model and the basic set of security policies specific to the secure software development infrastructure. This fact may diminish the determinancy of the secure software development analysis/synthesis process.

7. RELATED WORK

In the work [8], the authors proposed the secure software development lifecycle model applied by Microsoft. The key difference between the set of requirements proposed in this work and the one proposed in the work [8] is that the latter's results apply to a restricted software class (operating systems and database management system). Furthermore, the set of requirements offered in the work [8] is not traced to software lifecycle processes as per ISO 12207, making it somewhat difficult to implement software in the designer's current processes.

The work [3] offers software development requirements expressed in the form of confidence requirements and applied in information safety certification of software. The principal differences between the basic set of requirements proposed in this work and those proposed in the work [3] lie in the following:

- The set proposed in the work [3] is only applicable to software with security functions; the required security assignment cannot be written for software in which security functions have not been implemented;
- The set proposed in the work [3] has no completeness property: in particular, no provision is made for measures associated with static and dynamic analyses and personnel training.

In the works [11, 17], the authors offer a secure software development lifecycle model. The principal differences between the basic set of requirements proposed in this work and the one proposed in the work [11, 17] lie in that account is taken not only of development actions, but also of a valuator's potential actions to check designer-implemented processes for compliance with the specified requirements. Besides, the basic set of requirements offered in the works [11, 17] is not traced to software lifecycle processes as per ISO 12207, making it somewhat difficult to implement software in the already existing processes.

In the works [10], the authors offer measures to be carried out in a software development lifecycle, considering the need to conduct software certification in information security requirements and minimize the amount of software architecture-related vulnerabilities (for example, deficiencies in the subject identification and authentication mechanism). The results offered in this work can supplement the work [10] in updating the set of measures and requirements for valuators' actions.

8. CONCLUSION

We received the following main results in the course of the study:

1. We analysed the available controls and approaches aimed at minimizing the number of vulnerabilities in the software being developed, and their applicability during assessment of the software conformance to the information security requirements. The analysis established that there are currently a number of standards of various levels that are devoted to issues of secure software

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

development. However, there is no universal approach in a form of a regulatory document that ensures:

- Integration with the current standards that define the processes of the software lifecycle, for instance, defined by ISO 12207;
- Substantiated selection of a complex of secure software development controls;
- Assessment of conformance of the processes for creating software products to the requirements of secure software development, firstly, within the framework of mandatory certification of commercial information security software (during accreditation of production) and mandatory licensing of such types of activity as production of information security software.

2. Based on analysis, classification and summary of standards on secure software development we developed a basic set of 24 requirements for secure software development. The distinctive features of the developed basic set of requirements are:

- Cross references to the processes of the software lifecycle defined in ISO 12207, which ensures a simpler integration in the software development processes that already exist in the software developer (company);
- There are requirements to documented evidence of the requirement fulfilment and actions of the independent appraiser, which helps to determine the process of independent assessment of conformance of the software development processes to the requirements to secure software development, for instance, within the framework of the software certification.

3. We suggested a conceptual model for analysis and synthesis of secure software development controls, which ensures a possibility for substantiated selection of secure software development controls and is consistent with the Common Criteria standards. This allows for implementing the methods for analysis and synthesis of secure software development controls suggested within the framework of the model in companies that already have experience in certification of the developed software in accordance with ISO 15408 guidelines. The conceptual model for analysis and synthesis of secure software development controls was used during the development of the draft national standard GOST R that passed expert assessment within the framework of the Russian Technical Committee on Standardization (which includes 105 expert organizations).

REFERENCES:

[1] Duncan, B., Whittington, M. (2014). Compliance with standards, assurance and audit: does this equal security? In *Proceedings of the 7th International Conference on Security of Information and Networks* (Glasgow, UK, September 09-11, 2014). SIN '14. ACM, New York, NY, USA, (pp. 77-84). DOI:10.1145/2659651.2659711.

[2] Tillman, R.A. (2010). *Exploring Security Certification and Accreditation Using the Agile Software Development Lifecycle Process*. Ph.D. Dissertation. Capella University. AAI3402069.

[3] Merkow, M. S., Breithaupt, J. (2005). *Computer Security Assurance Using the Common Criteria.* Thomson Delmar Learning.

[4] Biró, M., Molnár, B. (2007). Synergies Between the Common Criteria and Process Improvement. *LNCS.* 4764 (31-45). DOI:10.1007/978-3-540-75381-0_4.

[5] De Win, B., Scandariato, R., Buyens, K., Grégoire, J., Joosen, W. (2009). On the Secure Software Development Process: CLASP, SDL and Touchpoints Compared. *Information and Software Technology.* 51, 7 (Jul. 2009), 1152-1171. DOI:10.1016/j.infsof.2008.01.010.

[6] Denning, P. J. (2012). Writing secure programs: an interview with Steve Lipner. *Ubiquity.* 2012, May, 1-10. DOI=http://dx.doi.org/10.1145/2213616.2213617.

[7] Geer, D. (2010). Are Companies Actually Using Secure Development Life Cycles? *Computer.* 43, 6 (Jun. 2010), 12-16. DOI=http://dx.doi.org/10.1109/MC.2010.159.

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

[8] Howard, M., Lipner, S. (2006). *The Security Development Lifecycle: A Process for Developing Demonstrably More Secure Software*. Microsoft Press.

[9] Kara, M. (2012). Review on Common Criteria as a Secure Software Development Model. *IJCSIT*. 4, 2 (Apr. 2012), 83-94. DOI:10.5121/ijcsit.2012.4207.

[10] Lee, M.-G., Sohn, H.-J., Kim, B.-M., Kim, J. B. (2015). A Study on Secure SDLC Specialized in Common Criteria. *ASTL*. 93 (11-23).

[11] McGraw, G. (2015). Software security and the building security in maturity model (BSIMM). *Journal of Computing Sciences in Colleges*. 30, 3 (Jan. 2015), 7-8.

[12] Mellado, D., Fernández-Medina, E., Piattini, M. (2008). Towards security requirements management for software product lines: A security domain requirements engineering process. *Comput. Stand. Interfaces*. 30, 6 (Aug. 2008), 361-371. DOI:10.1016/j.csi.2008.03.004.

[13] Essafi, M., Ghezala, H.B. 2014. Meta-Modeling Based Secure Software Development Processes. *IJSSE*. 5, 3 (Jul.-Sep. 2014), 56-74. DOI:10.4018/ijsse.2014070104.

[14] Fal', A.M. 2010. Standardization in information security management. *Cybernetics and Systems Analysis*. 46, 3 (May 2010), 512-515. DOI:10.1007/s10559-010-9227-9.

[15] Markov, A., Luchin, D., Rautkin, Y., Tsirlov, V. (2015). Evolution of a Radio Telecommunication Hardware-Software Certification Paradigm in Accordance with Information Security Requirements, In *Proceedings of the 11th International Siberian Conference on Control and Communications* (Omsk, Russia, May 21-23, 2015). SIBCON-2015. IEEE, 1-4. DOI:10.1109/SIBCON.2015.7147139.

[16] Shahriar, H. (2013). Security vulnerabilities and mitigation techniques of web applications. In *Proceedings of the 6th International Conference on Security of Information and Networks* (Aksaray, Turkey, November 26-28, 2013). SIN '13. ACM, New York, NY, USA, 459-459. DOI:10.1145/2523514.2523589.

[17] Viega, J., McGraw, G. (2011). *Building Secure Software: How to Avoid Security Problems the Right Way*. Addison-Wesley Professional.

[18] Ardi, S., Shahmehri, N. (2009). Introducing Vulnerability Awareness to Common Criteria's Security Targets. In *Proceedings of 4th International Conference on Software Engineering Advances* (Porto, Portugal, September 20-25, 2009). ICSEA '09. IEEE, 419-424. DOI:10.1109/ICSEA.2009.67.

[19] Buccafurri, F., Fotia, L., Furfaro, A., Garro, A., Giacalone, M., Tundis, A. (2015). An analytical processing approach to supporting cyber security compliance assessment. In *Proceedings of the 8th International Conference on Security of Information and Networks* (Sochi, Russia, September 8-10, 2015). SIN '15. ACM, New York, NY, USA, 46-53. DOI:10.1145/2799979.2800007.

[20] Seacord, R.C. (2014). *The Cert C Coding Standard: 98 Rules for Developing Safe, Reliable, and Secure Systems (2nd ed.)*. Addison-Wesley Professional.

[21] Barabanov, A., Markov, A., Fadin, A., Tsirlov, V., Shakhalov, I. (2015). Synthesis of Secure Software Development Controls. In *Proceedings of the 8th International Conference on Security of Information and Networks* (Sochi, Russia, September 8-10, 2015). SIN '15. ACM, New York, NY, USA, 93-97. DOI:10.1145/2799979.2799998.

[22] Stallinger, F., Neumann, R. (2012). From Software to Software System Products: An Add-on Process Reference Model for Enhancing ISO/IEC 12207 with Product Management and System-Level Reuse. In *Proceedings of the 2012 38th Euromicro Conference on Software Engineering and Advanced Applications* (Cesme, Izmir, Turkey, September 5-8, 2012). SEAA '12. IEEE Computer Society, Washington, DC, USA, 307-314. DOI:10.1109/SEAA.2012.41.

[23] Markov A.S., Tsirlov V.L. (2013). Opyt vyyavleniya uyazvimostey v zarubezhnykh programmnykh produktakh, *Voprosy kiberbezopasnosti* [Cybersecurity Issues]. 2013, 1(1) (Dec. 2013), 42-48. (In Russ.).