PSSS - Process to Support Software Security

Francisco José Barreto Nunes[1], Arnaldo Dias Belchior (*In Memorium*)

Mestrado em Informática Aplicada - Universidade de Fortaleza (UNIFOR)
Av. Washington Soares, 1321 CEP 60.811-341, Fortaleza – CE – Brasil

[1]fcojbn@yahoo.com.br

Abstract. *Software security becomes very important to organizations that depend on or whose customers demand software products that assure information integrity, availability, and confidentiality. Unfortunately, despite the investments made in process improvement according to Software Engineering practices, there is still no guarantee that the developed software products are immune to attacks or do not present security problems. This paper presents a software security approach based on a specialized process to help develop more secure software products, entitled Process to Support Software Security (PSSS). In addition, this paper presents the results of the PSSS's application in a software development project.*

1. Introduction

The growing need for software products to support business processes has motivated considerable research in the improvement of software development processes. In this sense, information security and security engineering increase their importance to become part of the business processes and the systems supporting these processes, in order to protect corporate assets and information. According to CERT (2007), software security defects are the main concerns that security professionals deal with.

More and more, software organizations are starting various projects and initiatives to develop more secure software products. For example, CLASP - Comprehensive, Lightweight Application Security Process – (2006) is a framework aimed at including security into a software development process.

OECD (2002) states the principle "Security design and implementation" as a characteristic of information systems. In fact, information security acts to protect information processed by information systems. This is achieved by maintaining the information confidential, available and correct.

Many specialists still think that a cryptographic function implemented in software to protect data integrity makes this software secure. Actually, this supposed secure software just implements a security characteristic and can not be considered secure. That is, security characteristics do not insure that software is secure [McGraw 2004].

Information security is related to many models and standards, like SSE-CMM (Systems Security Engineering – Capability Maturity Model) (2003), ISO/IEC 15408 (2005a, 2005b, 2005c), ISO/IEC 27002 (2005), and OCTAVE (The Operationally Critical Threat, Asset, and Vulnerability Evaluation) [Alberts *et al*. 2001]. This paper proposes the Process to Support Software Security (PSSS) based initially on the activities derived from these models and standards. In addition, this paper describes briefly the results of the application of this process in a software development project.

This paper is organized as follows: Section 2 describes briefly the security models and standards used to organize the activities of the PSSS; Section 3 describes the activities of the Process to Support Software Security; Section 4 explains in more detail the subprocess "Plan Security"; Section 5 analyzes the results of the application of the PSSS in a development project of an access control and audit software; and Section 6 presents the conclusions of this paper.

2. Information security models and standards

2.1 SSE-CMM

According to Anderson (2001), security engineering is about building systems to remain dependable in the face of malice, error, or mischance. As a discipline, it focuses on the tools, processes, and methods needed to design, implement, and test complete systems, and to adapt existing systems as their environment evolves.

SSE-CMM (Systems Security Engineering – Capability Maturity Model) (2003) ensures system security based on a framework that translates customer security needs into security products that satisfy the security requirements. SSE-CMM is a process reference model that describes security features of processes at different levels of maturity. The scope encompasses:

- The system security engineering activities for a secure product or a trusted system addressing the complete lifecycle of: concept definition, requirements analysis, design, development, integration, installation, operation, maintenance end decommissioning;
- Requirements for product developers, secure systems developers and integrators, organizations that provide computer security services and computer security engineering;
- Applies to all types and sizes of security engineering organizations from commercial to government and the academe.

SSE-CMM takes the view that security is pervasive across all engineering disciplines (e.g., systems, software and hardware) and defines components of the model to address such concerns.

According to Schumacher and Roedig (2001), SSE-CMM architecture treats security aspects such as privacy, confidentiality, integrity and availability. This architecture separates basic characteristics of a security engineering process from the institutionalization and management characteristics. Therefore, rather than regard SSE-CMM as a complex model to satisfy an overall system requiring full implementation, organizations can use the model as a pattern or benchmark for identification of current status across a set of information security processes and provide a rationale for security improvements. A previous version of SSE-CMM was adapted and became ISO/IEC 21827 (2002).

A restriction of the SSE-CMM is that it does not explain how to implement its process areas. Because of this characteristic, it is hard to understand and implement SSE-CMM.

2.2 OCTAVE

OCTAVE (The Operationally Critical Threat, Asset, and Vulnerability Evaluation) [Alberts *et al.* 2001] is a strategic planning and evaluation technique based on security

risks. The risks of the most critical assets are used to prioritize processes that need improvement and to elaborate a strategic orientation to mitigate these risks.

According to Alberts (2001), OCTAVE defines an approach to information security risk evaluations that is comprehensive, systematic, context driven, and self directed. The approach requires a small, interdisciplinary analysis team of business and information technology personnel from the organization to lead its evaluation process. That is, OCTAVE enables an organization's personnel to sort through the complex web of organizational and technological issues to understand and address its information security risks.

OCTAVE-S is a variation of the OCTAVE tailored to the limited means and unique constraints typically found in small organizations.

A restriction of OCTAVE is that it is a self-directed approach, meaning that people from an organization assume responsibility for setting the organization's security strategy. However, this could lead to security problems when these people have no security education or because they could have other responsibilities and are not entirely responsible for the security program.

2.3 ISO/IEC 15408

ISO/IEC 15408 (Evaluation Criteria for Information Technology Security) (2005a, 2005b, 2005c) presents a set of criteria to evaluate the security of products. This standard claims that a development process to produce secure software should include security in the development environment and security in the developed application. ISO/IEC 15408 is based on the Common Criteria (CC) (2005).

ISO/IEC 15408 is applied when it is necessary for the system to protect organizational assets. The security needs should be treated during all the development life cycle, from requirement management, functional specification, and project, to the final implementation in production environment. The secure development context of ISO/IEC 15408 is based on the execution of activities that describe how security requirements and specifications are derived when developing a software product.

Both the Common Criteria and ISO/IEC 15408 present a set of requirements that should be satisfied to make software more secure. The requirements are divided into: security functional requirements (2005b) and security assurance requirements (2005c). The former are a set of security characteristics that a software product can implement. The latter may function as actions to be executed during the development process to validate and certify that the software developed is secure because these actions were performed according to part 3 of ISO/IEC 15408 (2005c).

In addition, three similar difficulties were encountered in both the Common Criteria and ISO/IEC 15408: The requirements have complex correlations; they allow different interpretations; and there is no advice on how to fulfill these requirements during a software life cycle process.

A restriction of ISO/IEC 15408 is that it has its use restricted because of its complexity in implementing and assessing the security aspects of the software product. This standard requires a specialized knowledge which makes its use more expensive and time consuming. Another drawback is that ISO/IEC 15408 focuses individually on a software product and does not consider the interdependency among other systems and components.

2.4 ISO/IEC 27002

ISO/IEC 27002 (Code of Practice for Information Security Management) (2005) aims to preserve information confidentiality, integrity and availability in such type of business scenario. This is achieved through the implementation of security controls, including policies, practices, or processes. These controls ensure that defined security goals will be satisfied.

ISO/IEC 27002 states that it is necessary for an organization to identify its security requirements. This can be accomplished with the execution of risk assessments of the organization's assets by implementing vulnerability, threat and impact analysis.

A restriction of ISO/IEC 27002 is that it contains a vast number of security controls to be applied among different processes in any kind of organization. This could be seen as a weakness as this could lead to wrong interpretations. Another issue is that the standard does not explain how to best implement each security control.

Based on these models and standards, a mapping of similar activities was organized to draft the first PSSS. The SSE-CMM was the baseline of this mapping, sampled as shown in Table 1 below.

Table 1. Sample of the mapping of the security models and standards

SSE-CMM	ISO/IEC 15408	ISO/IEC 27002	OCTAVE
SSE-CMM : PA 04 – Assess Threat			
BP.04.02 – Identify Man-made Threats	Assess threats	___	- Describe areas of concern - Identify the threats to each critical asset by first mapping the areas of concern for each critical asset
SSE-CMM : PA 10 – Specify Security Needs			
BP.10.06 Define a consistent set of statements which define the protection to be implemented in the system	Specify functional security requirements	12.1 Identify information system security requirements	- Creates or refines the security requirements for the organization's critical assets - Create risk mitigation plans

3. Process to Support Software Security (PSSS)

The Process to Support Software Security was designed to follow the iterative and incremental life cycle approach which facilitates the coordination between the PSSS and any particular corporate development process. In order to use the PSSS with other life cycles, this would need validation.

There is no need to use all the activities of the PSSS. They can be adapted to function effectively within the organizational development process. It is an important aspect to have each activity as integrated as possible into the life cycle phases and one approach to reach this integration is to apply each activity in parallel with the phases.

In addition, the type and scope of a project define which activities to select to better suit the developed product. Besides that, the type of product defines the level of formalism and the rigors of evaluations to be implemented.

The PSSS identifies two important actors: the Security Engineer and the Security Auditor. The Security Engineer is responsible for the specialization of the

PSSS based on the objectives of the software development project and in connection with business plans and strategies. Another responsibility is to monitor if the project satisfies the security objectives.

The Security Auditor is responsible to evaluate whether the software development projects are done in compliance with the specialized PSSS. This person validates the effectiveness of the PSSS application, for example, in terms of the results of the activities and the artifacts developed. Preferably, the Security Auditor should not perform the activities of the software quality assurance team.

Both the Security Engineer and the Security Auditor should have experience, or at least knowledge, in security engineering, software engineering, software project management, and information security.

The PSSS considers 37 activities grouped in a set of 11 subprocesses (Figure 1). The subprocesses are described as follows:

3.1 Plan security

This subprocess assures that all information needed to plan the security of a project is defined and registered.

That is, the Security Engineer identifies the security objectives of the software to be developed, prepares the project security plan, and organizes information related to the project team.

The activities include: Developing security plan; Planning processing environments; Planning security incidents management.

3.2 Assess security vulnerability

This subprocess identifies and describes, for each iteration, the system security vulnerabilities related to the environment where the system would operate.

The Security Engineer is responsible to institute a vulnerability assessment method, to perform the identification of security vulnerabilities, and to analyze the identified vulnerabilities. In order to identify effectively the security vulnerabilities, the Security Engineer and the Software Engineer should organize interviews with selected users among executives, managers, and operational staff.

The activities include: Identifying security vulnerabilities; Analyzing identified security vulnerabilities.

3.3 Model security threat

This subprocess identifies and describes system security threats with their properties and characteristics based on the security vulnerabilities assessed previously.

Security threats can be defined as an event that compromise the normal behavior of software and, as a consequence, may have a negative impact in the organization. Threat modeling can be used to identify threats and make project decisions based on threats that can cause the major damage to the software.

The information about threats is necessary to develop strategies to reduce these threats. The strategies can influence the software development project, the coding, and test cases.

The Security Engineer is responsible to execute security threat identification and implement abuse cases and attack trees. Then, he selects an adequate approach to classify these threats and organizes interviews with defined users among executives, managers, and operational staff to develop strategies to reduce the impact of these threats.

The activities include: Identifying security threats; Classifying security threats; Developing strategies to reduce security threats.

3.4 Assess security impact

This subprocess identifies and describes relevant system security impacts and defines the probability of their causes based on the security vulnerabilities and threats assessed previously.

Security impacts can be tangible, such as fines, or intangible, like loss of reputation. Impact is defined as a consequence of an undesirable incident that can be caused deliberately or accidentally and affects the software. The consequences can result in destruction or damage of software artifacts or even in loss of confidentiality, integrity, or availability.

The Security Engineer, Software Engineer, and users (customer) are responsible to prioritize the critical activities influenced by the software. Security Engineer and Software Engineer review software's security artifacts.

Based on the previous information and information about security vulnerabilities and threats, the Security Engineer and users identify security impacts from unwanted incidents and detailed information about these impacts.

The activities include: Treating critical activities for security; Reviewing system security artifacts; Identifying and describing security impacts.

3.5 Assess security risk

This subprocess analyzes the security risks of the developing system by identifying the security exposure, the risk caused by this exposure, and the priorities of these risks based on the security vulnerabilities, threats and impacts assessed previously. The risk assessment verifies the exposures which can prevent the software to meet its objectives.

The Security Engineer and Software Engineer, with the help of the user, identify security exposure by evaluating and prioritizing security risks.

The activities include: Identifying security exposure; Assessing security exposure risk; Prioritizing security risks.

3.6 Specify security needs

This subprocess specifies the security needs of the system according to stakeholders' security needs.

The Security Engineer, with the help of the user and of the customer, is responsible for understanding customer's security needs and for developing a high-level security view of the software. This high-level view helps to define the security requirements. Finally, the Software Engineer mediates between customers and Security Engineer to obtain agreement about the security requirements.

The activities include: Understanding customer security needs; Capturing a high-level security view of the system; Defining security requirements; Obtaining agreement about security requirements.

3.7 Provide security information

This subprocess provides system architects, designers, implementers, or users with any security information needed to perform their work.

Security information would be considered kind of information which has impact on, is necessary to support, or helps members of a software security project.

Additional security information is also provided to examine software security problems against the defined security objectives, to make team members understand the PSSS, and to guarantee that the software product implements the security requirements.

The Security Engineer acts as a security mentor to the project team identifying necessary information and making it available. For example, architects could need information about security in Web services. In this case, the Security Engineer interacts with Software Architects to give the information.

The activities include: Understanding information security needs; Identifying security constraints and considerations; Providing security alternatives; Providing support requirements.

3.8 Verify and validate security

This subprocess assures that software solutions are verified and validated according to their designed security goals. That is, the subprocess tries to guarantee that solutions are verified and validated in relation to the security requirements, architecture, project and customer security needs based on observation, demonstration, analysis and test. Because of this approach, customers are more confident that the software solutions implement effectively the security requirements and therefore satisfy their security needs.

Verifying security ensures that software satisfies specified security requirements. Validating security demonstrates that software accomplishes its intended security requirements when placed in its production or operational environment.

The Security Engineer, with the help of the Software Engineer, defines security verification and validation approach that involves plan elaboration, scope, depth, and tests.

Then, the Security Engineer, with the help of the quality assurance team, performs the security verification and validation, reviews and communicates the results.

The Security Auditor assesses the security verification and validation to check whether the activities are being performed correctly.

The activities include: Defining security verification and validation approach; Performing security verification; Performing security validation; Reviewing and communicating security verification and validation results.

3.9 Manage security

This subprocess controls the activities needed to organize and to keep the security mechanisms to the software development project, as well as to manage the control implementation for new functions.

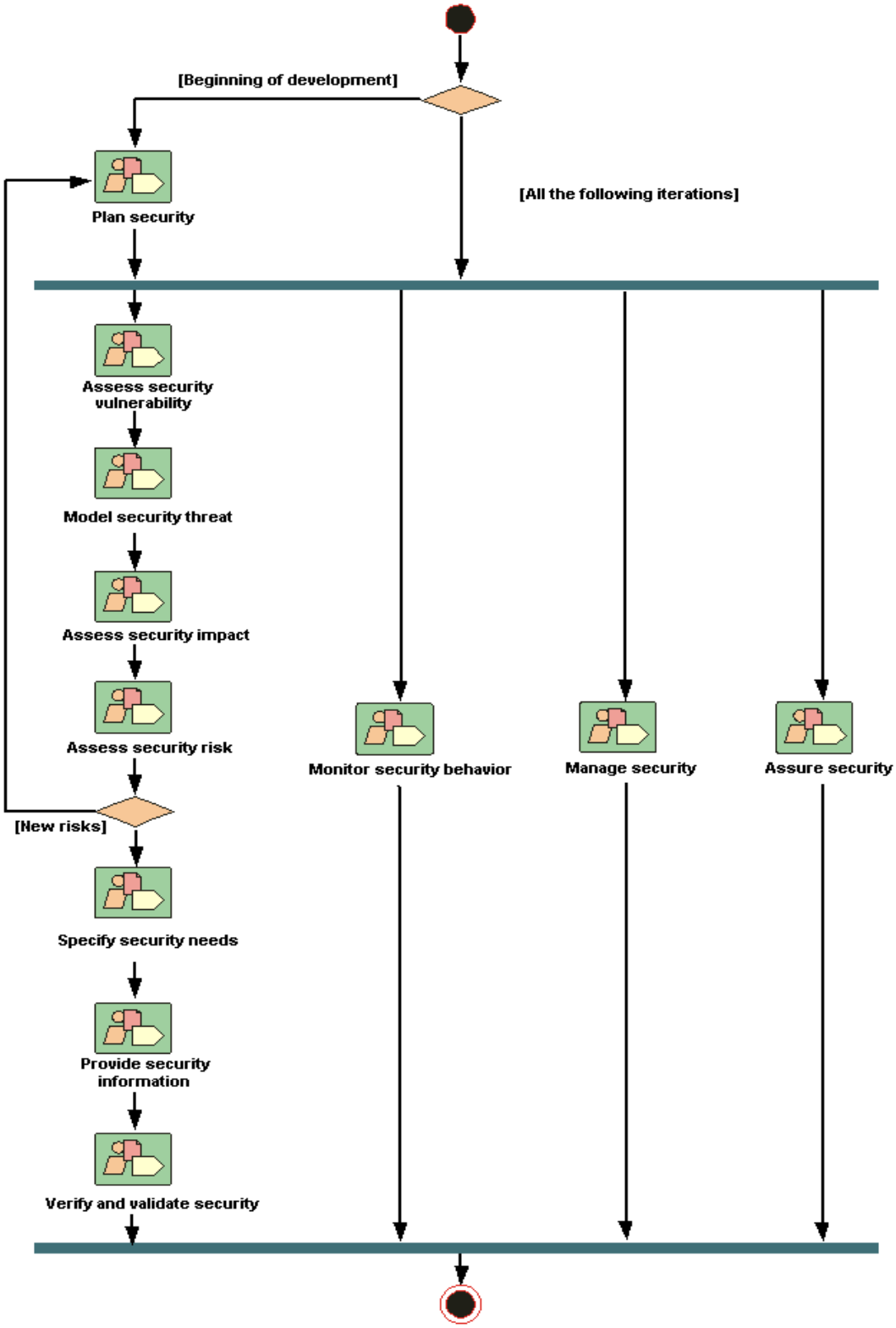

Figure 1. Process to support software security

Another purpose of this subprocess is to define how the security management will be organized, which includes security educational programs and security training. The security services and mechanisms for the software development project are detailed. In addition, this subprocess addresses the issues identified during the subprocess "Verify and validate security".

The Security Engineer deals with and controls additional security services and system components. The Security Engineer identifies training needs and educational programs about information security and about the process to support software security. Finally, he manages the implementation of security controls in the software being developed. The Project Manager helps the Security Engineer with the management of all the activities of the PSSS.

The activities include: Managing security services and components; Managing security training and education programs; Managing the implementation of security controls.

3.10 Monitor security behavior

This subprocess monitors the system developed and already in use to identify whether the security features defined in the project are achieved.

In this subprocess, the internal and external environments are monitored in relation to the factors that can impact software security. The main objectives are: Internal and external security events are detected and supervised; Incidents are treated according to the incident management plan; and Changes in the security circumstances are identified and handled according to security objectives.

The Security Engineer, with the help of the Software Engineer, is responsible for the following activities: Analysis of events with security impact; Identification and preparation of the incidents response; Monitoring changes in environments, in security vulnerabilities, threats, and risks, and in their characteristics; and Reviewing software security behavior to identify necessary changes.

The Security Auditor assesses the activities described below to identify irregularities and problems. Besides that, he is responsible for (1) Reassessment of the changes in environments and in security vulnerabilities, threats, and risks; (2) Performing security audits.

The activities include: Analyzing events with security impact; Identifying and preparing the answer to relevant security incidents; Monitoring changes in security threats, vulnerabilities, impacts, risks, and environment; Reviewing system security condition to identify necessary changes; Performing security audit.

3.11 Assure security

This subprocess defines a set of activities which can be applied to guarantee that the security of the software product is achieved.

The stakeholders should receive information to assure that their expectations are satisfied in relation to the effective application of the PSSS and the security of the software being developed.

Therefore, this subprocess aims to assure that effective controls are defined and implemented to guarantee the correct protection of critical artifacts (information, function, database tables, etc).

The Security Engineer and the Software Engineer, with the help of the customer, should develop a strategy to guarantee the maintenance of security assurance. This strategy promotes the effective performance of this subprocess.

The Security Auditor executes an impact analysis based on security changes to assure that no change compromises software security. Finally, the Software Engineer and the Security Auditor control the security assurance evidences that confirm this maintenance.

The activities include: Defining security assurance strategy; Performing security change impact analysis; Controlling security assurance evidences.

In the next section, the subprocess Plan Security will be exemplified.

4. Subprocess "Plan Security"

The purpose of the subprocess Plan Security (Figure 2) is to define, to establish, and to register the necessary information to plan the security in a software development project. Examples of information to plan the security include: project scope, objectives, activities to be used, project team, and environment.

The information security objectives and plans should be elaborated (or refined) and a security team (as the instantiation of the PSSS) organized.

The Security Engineer is responsible, with the help of the project team, for the definition of the security vision to be established for the project. The Security Engineer may be also a software security consultant for the stakeholders. Each one of the activities is described below.

4.1 Develop security plan

The purpose of this activity is to define the security plan and to identify project coordination mechanisms.

The business security objectives are identified from information security and software development documents, among others, and organized in the artifact "Organizational security assets". Security objectives and strategies of the project are selected from this artifact and from the software development plan. Therefore, objectives and scope of the PSSS according to the project are defined.

The project's security team should be structured which includes the Security Engineer, the Security Auditor and other specialists whose experience could help the application of the PSSS. Responsibilities and roles between the software development team and the security team should be agreed to guarantee that the communication during the project flows correctly.

The security team is responsible for educating and guiding other project teams to apply the PSSS, perform its selected activities, and integrate the PSSS with the organizational software development process.

The Project Manager helps the other actors to understand the project to assure that the defined objectives and coordination mechanisms correspond to the project characteristics.

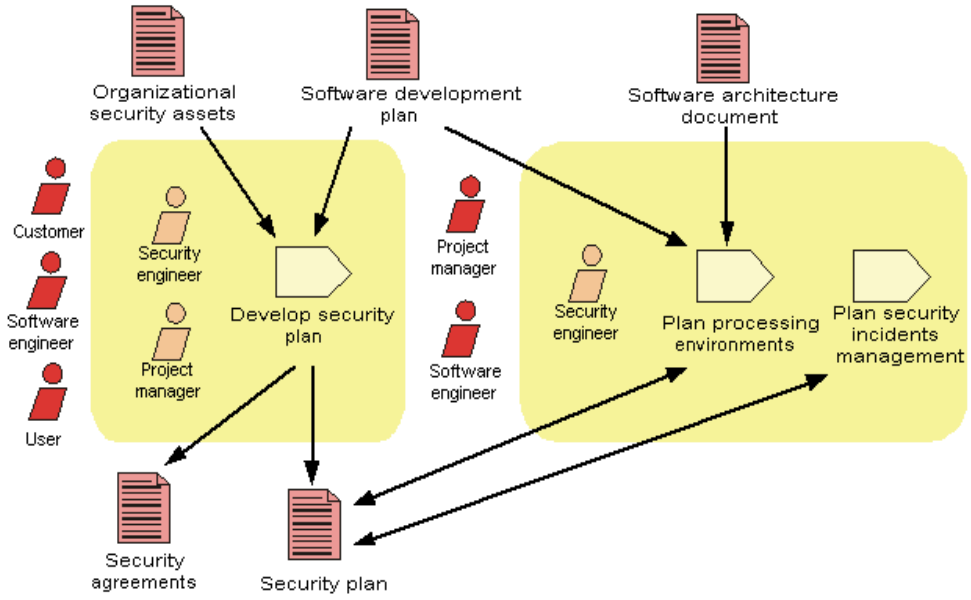

Figure 2. Subprocess "Plan Security"

4.2 Plan processing environments

The purpose of this activity is to identify and analyze the development, test, and environments of production. The separation level needed among these environments is evaluated to prevent from operational and security problems.

4.3 Plan security incidents management

The purpose of this activity is to analyze the project to verify the plan to manage security incidents.

This activity defines the project's incident management plan in order to have a fast, effective and coordinated security incidents response.

In the following section, the results of the application of the process to support software security will be presented.

5. Application of the PSSS

The PSSS was applied in a large sanitation public company with more than four million customers. This company has a small development team involved with java development.

The application occurred with a small set of activities. The selection of these activities was due to time constraint of this project, i.e.: 4 (four) weeks, and these activities were selected based on the characteristics of the project and on the experience of the software engineering team. This activity selection approach contributed to make the application of the PSSS costless to the company.

The PSSS was applied in a software development project of an access control and audit web-based system that, among other functions, defines and controls user access and registers the actions of these users. The system's functionalities were formalized among 9 (nine) use cases.

Firstly, each activity was described and explained to the sponsor, the Chief Information Officer, and the Chief Financial Officer. Then, the activities were evaluated to verify if they could be really applied to the project and, afterwards, a simple version of a security plan was prepared.

Next, the Security Engineer (new role created in the company) and the Software Engineer identified and analyzed security vulnerabilities and identified security threats. The information related to security vulnerabilities and threats helped to understand and select security needs. With these security needs and with the help of abuse cases (Figure 3) and attack trees (Figure 4), it was possible to identify a set of security requirements:

- Prevent the creation of unsafe passwords: the system must assure that every password creation and password change follows defined password creation criteria;
- Prevent wrong system access: the system must provide a secure user access with authentication and validation;
- Prevent the change of registered audit information: the information related to the actions of users and administrators must be registered for consultation and this information must not be changed or deleted.

Finally, after design and implementation, the security and software engineers verified whether the security requirements were implemented in the final software according to the activity "Perform security verification". However, only a small portion of the security tests was done because of project time constraints. For example, penetration tests were not executed. The results obtained from the security verification and validation indicated the end of the PSSS's application.

Figure 3. Example of an abuse case of the access control and audit system

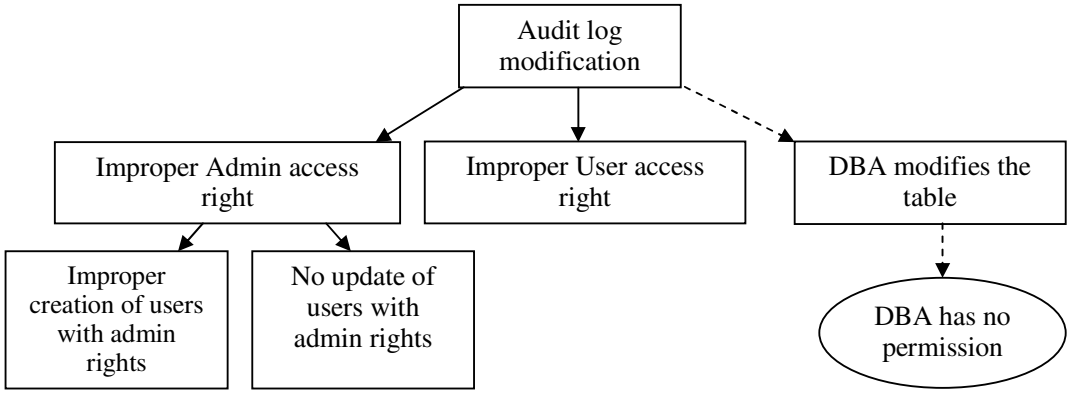

Figure 4. Example of an attack tree of the access control and audit system

The following points were noticed after the initial application and are worth mentioning:

- In addition to the previously selected activities, it was necessary to execute informally other activities, as "Performing security validation" and

"Performing security verification", in order to obtain a satisfactory result in this application;

- Some performed activities, as "Developing security plan" and "Identifying security vulnerabilities", were adapted to improve its effectiveness to fit in the project's peculiarities;
- An effective method to identify assets is an important aspect for a successful application of the PSSS, in order to facilitate the assessment of vulnerabilities and threats;
- The inexperience to perform security vulnerability and threat identification demanded more time to perform these activities;
- There was not an organizational knowledge base with system problems and flaws to help an effective security vulnerability and threat identification;
- It was necessary to prepare formal training in java secure programming.

Although the PSSS had been received enthusiastically by the sponsors, high management, project manager, developers and users, it presented some problems during the application. The main problems included:

- Identification and understanding of software assets (artifacts);
- Lack of knowledge to implement in its entirety the activities related to threat modeling;
- Insufficient time for the teams to get used to the PSSS and its activities;
- Need for additional resources to implement effectively the PSSS.

The main advantages gained by applying and following the PSSS were:

- Assurance that security was considered during the system development through elaboration of security activities and artifacts, such as attack trees and abuse case, and that potential security vulnerabilities, threats and risks would be treated;
- Identification and definition of security requirements based on a set of security assessments to protect the system against security problems. One example of this problem is loss of information integrity were avoided because prevention controls were developed against creation of unsafe password or unauthorized audit log modification;
- Assurance that the limited project resources were effectively applied based on security assessments and according to the major negative security impacts.

The security status of the system and user satisfaction increased not only because the security requirements were implemented but also because the customer and users have known that the software was being developed with security consideration and precautions in the form of an organized software security development process.

6. Conclusion and future work

This research aims to consolidate the proposition of a set of information security activities divided among subprocesses which were derived from SSE-CMM, ISO/IEC 15408, ISO/IEC 27002, and OCTAVE and selected to form the Process to Support Software Security (PSSS).

Although there are well organized software development processes based on maturity models or international standards, the approach and ability to develop secure

software have not been achieved yet. That is, they still lack adequate support for security.

The proposition of a software development process formed with security activities could help in the production of more secure software as it protects the confidentiality, integrity, and availability of processed and stored information, satisfying the growing customer demand for security in software products. According to Howard (2002), application security should be designed and incorporated into the products since the beginning of the development process.

It was verified that information security, among SSE-CMM, OCTAVE, ISO/IEC 15408 and ISO/IEC 27002, is not directly related to a software development process. This fact increases the importance of establishing a set of information security activities forming a process to help developing more secure software and the importance of discussing security in software products in a software engineering context.

The PSSS could also be seen as a security engineering approach to improve the effectiveness of software security projects. The development and organization of the PSSS tried to avoid common problems and restrictions of these information security models and standards so as to not repeat them or limit their influence on it.

Some contributions of the PSSS are:

i. Increase the awareness about the importance of including security into software development by applying a self-oriented process based on known security approaches;
ii. Stimulate the assessment of security vulnerability, threat, impact and risk in every phase of the development process according to defined security activities;
iii. Show the necessity and the importance of making security requirements assessment based on security vulnerability, threat, impact and risk information;
iv. Reinforce the importance of executing security tests to validate and verify software security as a continuous activity that depends on security requirements elicitation;
v. Express the necessity to formalize a process to assure that the established security was completed and accepted; and
vi. Permit proactively the identification and correction of software security problems.

Despite these advantages, the implementation of the PSSS may initially increase the need for more resources and investments which can vary depending on the project's specifications. However, the utilization of the PSSS is recommended in comparison to best practices and practical experiences because their conceptual bases are incipient, they do not consider aspects of analysis and design phases satisfactorily, and they fail to perform proactive actions in accordance with security engineering principles.

As a final comment, it appears that the utilization of the Process to Support Software Security should be considered to develop more secure software because this process includes important characteristics from security models like ISO/IEC 15408 (Common Criteria) and SSE-CMM. Moreover, the PSSS's capability to be specialized facilitates its utilization in different companies and its implementation among different projects.

Further work will be done on applying the PSSS in a bigger software development project to perform activities from all the subprocesses and to obtain new results to make the PSSS more effective and consistent.

References

Alberts, C. *et al*. (2001) "*OCTAVE - The Operationally Critical Threat, Asset, and Vulnerability Evaluation*", Carnegie Mellon – Software Engineering Institute. Available at: www.cert.org/octave.

Anderson, R. (2001), *Security Engineering: A Guide to Building Dependable Distributed Systems*. John Wiley and Sons.

CERT. (2007), *Coordination Center Statistics*. Available at: www.cert.org/stats/cert_stats.html.

CLASP. (2006), *Comprehensive, Lightweight Application Security Process*. Version 1.2. Available at: www.owasp.org/index.php/owasp_clasp_project.

Common Criteria (2005), Version 2.3, August 2005. Available at: http://www.commoncriteriaportal.org.

Howard, M.; LeBlanc D. (2002), Writing Secure Code, 2nd edition. Microsoft Press.

ISO/IEC 15408-1. (2005a) *Information technology – Security techniques – Evaluation criteria for IT security – Part 1: Introduction and general model.*

ISO/IEC 15408-2. (2005b) *Information technology – Security techniques – Evaluation criteria for IT security – Part 2: Security functional requirements.*

ISO/IEC 15408-3. (2005c) *Information technology – Security techniques – Evaluation criteria for IT security – Part 3: Security assurance requirements.*

ISO/IEC 21827. (2002) *Information technology - Systems Security Engineering - Capability Maturity Model*.

ISO/IEC 27002. *Information technology – Security technical - Code of practice for information security management*. 2005.

McGraw, Gary (2004), *Software Security*, IEEE Security and Privacy, 2(2): 80-83, 2004.

OECD. (2002) Organisation for Economic Co-operation and Development. *Guidelines for the Security of Information Systems and Networks: Towards a Culture of Security*, page 13, Principle 7, Security design and implementation. Available at: www.oecd.org.

Schumacher, M., and Roedig, U. (2001), "Security engineering with patterns", In Proceedings of the PLoP conference on Pattern Languages of Programs (Illinois, The USA, September 11 – 15, 2001).

SSE-CMM. (2003) *System Security Engineering – Capability Maturity Model*, Version 3. Available at: www.sse-cmm.org.