Software Lifecycle Activities to Improve Security Into Medical Device Applications

Diogo C. Rispoli , Lourdes M. Brasil
Graduate in Biomedical Engineering
University of Brasília at Gama
Brasília, Brazil
e-mail: drispoli@gmail.com, lmbrasil@gmail.com

Vinicius C. Rispoli
Faculty of Engineering at Gama
University of Brasília
Brasília, Brazil
e-mail: vrispoli@unb.br

Paula G. Fernandes
Computer Science Department
University of Brasilia
Brasília, Brazil
e-mail: paulag6@gmail.com

Abstract—**This work proposes a methodology to include into Medical Software Development Lifecycle activities that helps improve security. The methodology uses assessment techniques and methods, applied to each phase of software lifecycle, that address security concerns and help to improve software quality. As a result, a partial analysis using the methodology proposed was performed in medical software at development stage to help reduce its gap between safety and security requirements.**

Keywords-information security; security software; hackers; medical software lifecycle; security risks.

I. INTRODUCTION

In a scenario of constant technological evolution, the demand for solutions in medical field is constant. Nowadays, you can find free software available on the internet which collect vital information of individuals and can be installed on mobile devices, such as smartphones and tablet computers [1]. Despite this speed, standards focused on medical field do not present models that make possible to assess problems associated with common security vulnerabilities that may appear in the software development cycle.

Standards as ISO / IEC 62304:2007 [2] dealing with the medical equipment software development, although recent, do not handle with these new technological perspectives. On the other hand, ISO 27799:2008 [3] deals with security concerns of health information systems, but it does not address solutions related to secure software development, compared to the present moment.

This very moment of technology effervescence opens doors for hackers to exploit and promote invasions as the attack on an insulin pump documented and presented in [4]. Based on a simple technique that combines programming skills and basic electronics, the hacker undertakes an attack on an insulin pump, used by himself, in order to demonstrate the innocent perspective that these devices are built. As a final result, he can apply a lethal dosage of insulin breaking the authentication security required by the equipment wireless communication.

Remembering that this was not the first case of attack documented on medical devices. In 2008, a U.S. team of researchers published a paper that showed an attack on a pacemaker, which also exposed security flaws related to wireless equipments [5].

Evidently, there is a rush in adopting standards, and actions, to ensure the security of the software built for medical devices. The exploitation of vulnerabilities in medical equipment can lead to death or serious injury, fraud, unauthorized disclosure of information, theft, and other attacks. For this reason, it is necessary to ensure that information security requirements (integrity, confidentiality, availability and non-repudiation of data collected) are met as well as ensuring that software vulnerabilities are not included in these devices during its development lifecycle.

This article will discuss requirements for improving the security of medical applications based on risk assessment of information security and the correlation of requirements for software security standards and their mitigation techniques related to safety in life support. As a result of this work, activities, also known as touch points, will be shown and assessed through a software development lifecycle helping to ensure the security requirements needed to consider software secure and safe.

This document is divided in six sections. In the next section, the relationship between risk perspective from safety and security views will be discussed. In section three, we will present the importance of software lifecycle and the incorporation of security activities into software construction. In the fourth section, the assessed software and its characteristics will be presented. Software assessment against the methodology proposes will be shown the in fifth section. And, in last section, will be discussed the assessment results.

II. SECURITY RISKS

In order to associate issues that are seemingly disconnected, it is important to observe how software security aspects are linked to medical devices construction. Under the perspective of the software, the risks are paths through the application where attackers (hackers) can disrupt business or organizations [10].

Observing this look from the perspective of the equipment, the risk (or level of concern) is an estimate of the injury severity which equipment can inflict or allow, directly or indirectly, in a patient or operator, as a result of device failure, design flaws, or because of the device employment for its intended use [9].

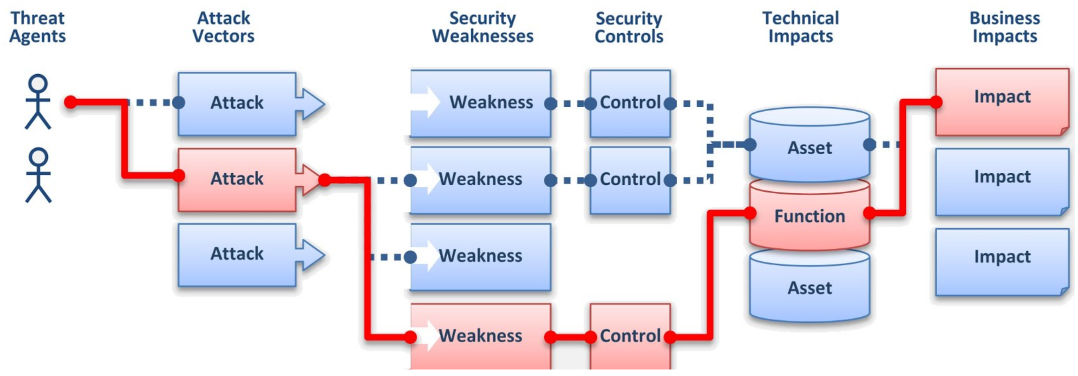


Figure 1. Applications Security Risks [10].

Objectively, these two views are very close because risks are directly related to software failures or weaknesses in its control mechanisms. Its natural consequence is the subversion and many kinds of damage, primarily damage to life, but also financial and corporate image caused by malicious people.

Threat agents can use several paths over application in order to attack organizations. These paths are through exploration of security weaknesses to bypass security controls and cause technical and business impacts, as it can be seen in Figure 1.

From this perspective, raising, mapping and balancing risks, flaws and vulnerabilities introduced by problems in the construction of medical software becomes exhausting, ineffective and away from the current technological reality. There are many patterns as you can see in Figure 2, dealing directly (such as IEC 62304:2006) or indirectly with software development lifecycle and its associated security risks [11].

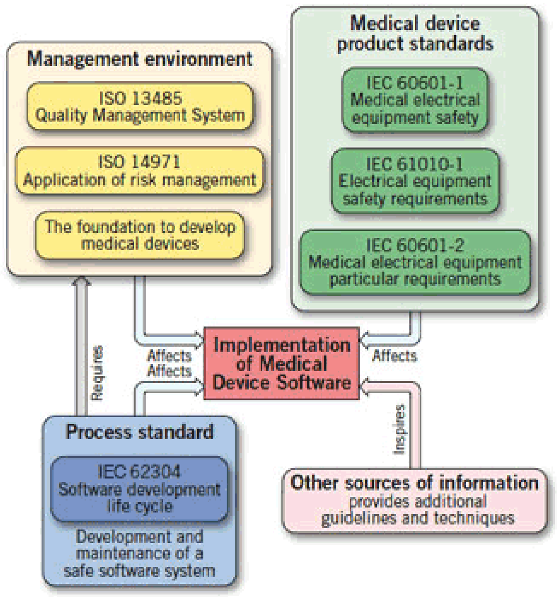

Management environment
ISO 13485 Quality Management System
ISO 14971 Application of risk management
The foundation to develop medical devices
Medical device product standards
IEC 60601-1 Medical electrical equipment safety
IEC 61010-1 Electrical equipment safety requirements
IEC 60601-2 Medical electrical equipment particular requirements
Implementation of Medical Device Software
Requires
Affects
Affects
Affects
Inspires
Process standard
IEC 62304 Software development life cycle
Development and maintenance of a safe software system
Other sources of information provides additional guidelines and techniques

Figure 2. ISO/IEC 63304 and its relation to other standards [11].

However, there is a lack of methodologies that address mitigation aspects for exploitable vulnerabilities in software. It is important notice that IEC 62304:2006 address security as concern that manufacturers shall include in software requirements [2].

The ISO 27799:2008, which concerns to medical information systems, do not treat or address solutions related to the process of building secure software. Notwithstanding, this standard imposes requirements on the operation of informational systems as secure authentication, authorization, accountability, use of encryption, secure information communication and protection against code injection. These are relevant aspects where the software is the leading actor or an important supporting actor [3].

Observing the processes of quality assurance employed in medical applications, the aspects of validation and verification are only concerned with functional requirements of the software [10].

So, it is necessary list interactions with the lifecycle of the software that shows a path to perform penetration tests and audits, raise non-functional requirements for safe operation and deployment of applications, including risk analysis of vulnerabilities in software design, protect applications against command injection flaws and buffer overflow, properly handle errors and exceptions and logging sanitized records (after removing sensitive information) of users activity in the equipment operation [3, 8, 10, 12].

It is also important to design efficient mechanisms for authentication, secure session management, user authorization, authenticated encryption for secure transmission, storage of collected data and records of patients with the goal of increasing the guarantee of the safety and quality of information systems health and its related applications, whose assets are devices and their associated software [3, 10].

III. Software Development Lifecycle and Security Concerns

Lifecycle models organize development software activities and provide a framework to monitor and control a building software project and its future operation. Without a model is difficult to say the exact moment of project's development or validation phase and how or which situations control activities must be applied [7, 8].

Despite *Quality Systems Regulations (QSRs)* do not establish a specific lifecycle model for medical software, regulatory standards state that a model adoption is important and it should contain at least some phases like quality planning, requirements management, software project specification, coding, testing, installation, operation, support and maintenance [7, 9].

Development of checklists with controls to be applied can aid incorporation of secure coding practices throughout the construction of medical software. The use of security software techniques does not necessarily increase the cost of its development lifecycle, in order to correct problems and failures of this nature cost more after application development finishes [8].

Adoption of security techniques is expected since medical software is able to run into smartphones and other mobile devices, for example, and all information collected and transmitted by those devices are sensitive and confidential.

Software development lifecycle is part of project controls and these controls are needed to reduce flaws insertion in medical device [7]. So, to help mitigate medical software vulnerabilities problem is essential to indicate what activities must be implemented between software lifecycle phases. These activities are related to the identification, development and validation of techniques that difficult vulnerabilities exploitation in software operation.

An interesting way to improve software security and quality is perform security activities through software lifecycle. Those activities are responsible to manage security concerns and must be applied inside lifecycle phases instead deal with security concerns only at requirement phase, as suggested by IEC 62304:2006 [2]. A correlation between phases and activities can be seen in Figure 3; they were described for general software projects in [8].

It is important to notice that there is no specific methodology to use the described security touch points. They can be applied in every kind of software development lifecycle methodology [8].

A. A brief description of each touch point

The touch points are described as follows [8]:

Abuse Cases – Build abuse cases is relevant to do relationship between problems and risk analysis. At that moment is important observe if some attack pattern fits the system or software requirements. This is a good moment to model vulnerability scenarios that could be exploited in Code Review Phase and Penetration Testing Phase.

Security Requirements – Security requirements must cover functional security, safety requirements, raised abuse cases and attack patterns. In that phase every software security necessity must be mapped to ensure the correct implementation. A good example for security requirements is the correct use of cryptography to protect critical data.

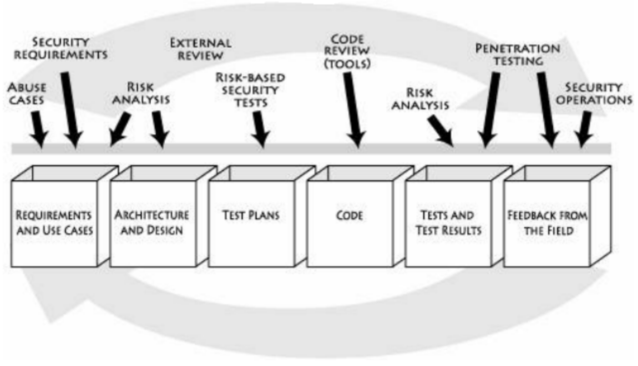


Figure 3. Security touch points inside a lifecycle [8].

Architectural Risk Analysis – Completing risk analysis oriented by ISO 14971. This analysis is a small part of a Risk Management Process that every Manufacturer must apply complying with ISO 14971, according to [2].

Risk Based Security Tests – The testing strategy must cover at least to major topics: test security requirements with standard functional testing techniques and risk-based security testing build from abuses cases and attack patterns.

Code Review – After codification, and before testing phase, the code review analysis is a good activity to ensure the security requirements were well implemented and the vulnerabilities listed in abuses cases analysis are outside the software. The code review can be automatic or manual and each strategy has pros and cons. Automated tools do not enforces all scenarios; some will require manual assessment [14].

Penetration Testing – This is a set of techniques and tools used together to test the software application dynamically against design flaws or vulnerabilities. This activity is important to guarantee that the software or its infrastructure do not have any potential problem that can be exploited in a particular way and change its behavior on the fly.

Security Operations – It is very important to log the user activity into software system usage. Even more important is to maintain that data in a correct and protected manner, to ensure that the attacker or attack activities can be tracked down after the attack attempt.

IV. SOFTWARE ASSESSED

The assessed medical device is responsible to monitor vital signs from a patient and send collected information to an Android smartphone. This system, showed as a diagram at Figure 4, is divided into a Body Sensor Network (Figure 5a), composed by a sensor set that monitor vital signs, a Coordinator sensor that collects information from body sensors in a regular basis and re-send that data to the smartphone and the Monitor software (Figure 5b) that evaluate patient conditions time to time. For this software /

equipment no injury is possible, arranging it into Class A classification, according to [2].

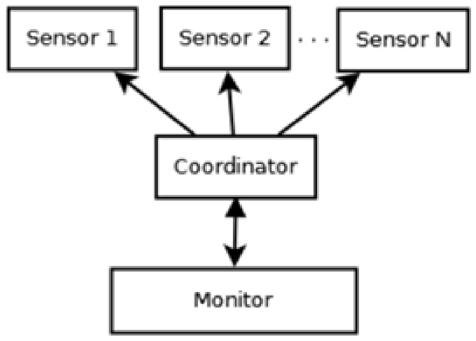


Figure 4. Monitor System Diagram.

This medical device is developed as a research project of the Software Engineer Group from Computer Science Department at University of Brasilia and was provided as a courtesy for this assessment. The research group responsible for developing the monitor system is not the same group that performed the software analysis. Notice that the only part assessed in this work is the Monitor software. Mechanical parts, sensors and smartphone hardware are not part of that analysis. Monitor software was developed in Java Language to run in Android devices.

This software uses the *Software Product Line (SPL)* methodology to build reusable components. In *SPL*, each product is a different piece of software that has some common artifacts in its structure [17]. In medical area, the use of *SPL* methodology brings some problematic issues related to validation and verification of safety characteristics. So, the research team [17] built Monitor software to verify the use of a parametric validation checking model to ensure safety properties (availability, reliability, security, integrity and maintainability). It was done because all medical device software must have dependable and reliable characteristics to guarantee safety.

This device and its related software were a good candidate to security evaluation since the software was in early development state and uses an unusual development methodology for medical devices. It is especially interesting to see if security activities really fit into a new development methodology or perspective.

V. Analysis of the Software Based on the Methodology Proposed

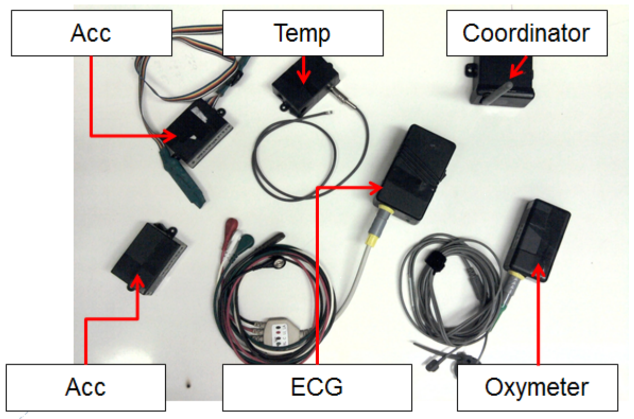
The analyzed software was not plan or built with any security touch point in mind. To help improve the software security and safety was performed an evaluation to propose and add touch point activities into software lifecycle, especially into building steps. Those touch points could be added into software lifecycle at any time, but it is better to do it when the software contains those activities from the scratch.

There are some steps to assessment take place. These steps can be related with one or more touch points each time and they were performed to track the assessed software into a security lifecycle.

Just for the record, safety practices listed at ISO 62304:2006 and other standards will not be ignored here but overlapped by security perspectives. It will be added at software process to increase safety and establish security. For example, risk analysis, abuse cases and risk-based tests are already present in safety related processes and this work will bring security concerns to these activities.

Abuse cases are related with vulnerabilities and flaws. For this analysis were defined SQL injection vulnerability and authentication and authorization problems as abuse cases. SQL injection, for example, could reveal validation problems in application. That is a common vulnerability in software [10, 14, 15] and must be mitigated. Authentication and authorization problems could show problems related to software design flaws [15].

Risks, in a security perspective, are directly related to software failures or vulnerabilities. The risk for SQL injection vulnerability is information disclosure and for authentication and authorization problems are non-legitimate user accessing and exploring the application. The risk-based security tests will be related to the risks specified. In code phase, these risks must be mitigated to ensure no path for exploitation.



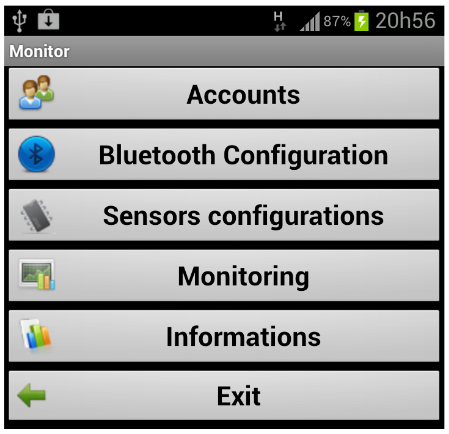


Figure 5. (a) Body Sensor Network (*left*). (b) Monitor software interface (*right*).

Code review is an important control strategy. This methodology comprises, at least, the following elements: Track user-controllable entry point data and review source code responsible for process it, search evidences to ensure that there is no vulnerability related to risks in source code and look for known patterns for common vulnerabilities and perform a line-by-line review of risky code to understand application logic and flaws that may exist [14, 16].

The code review phase could use tools, but it is necessary keep in mind that tools does not do all work. Manual review is always required.

Problems related with abuse cases and with risks specified above were found in Monitor software source code when performed a detailed code review. Field validation and authentication controls are not properly implemented. Examples of vulnerabilities found in source code review are shown in Table 1.

TABLE I. SOME PROBLEMS FOUND IN CODE REVIEW

#	Vulnerabilities		
	Flaws	***Class Name***	***Line Number***
1	Logging of user activity	Global (Many Classes)	N/A
2	No validation on input field	AccountMaintainActivity.java	142
3	Persistent Command Injection	UserDAO.java	119

To confirm that problems found in source code review could really be exploited, a penetration test must be performed. There are, at least, three phases involved in penetration testing: test preparation, test and test analysis as shown in Figure 6.

First phase is related to scope, objectives, timing and duration of the test. All legal agreements must be arranged during this phase. Second phase is considered the bulk of penetration test process. This phase involves application information gathering, vulnerability analysis and exploits. Results are investigated and analyzed in the last phase. The final report generated must be comprehensive and systematic [18].

Security operation is concerned with platform problems that could happen while software is working. Monitor software must be configured following Android Platform security specifications and requirements, as show in [13]. Examples of described requirements are data protection, cryptographic practicalities and use of protected communication channels.

This work is not confirming source code review with a penetration test since application is on early development stage. As soon as Monitor software development starts follow a security development plan, regular dynamic evaluation will be performed as soon as software becomes mature.

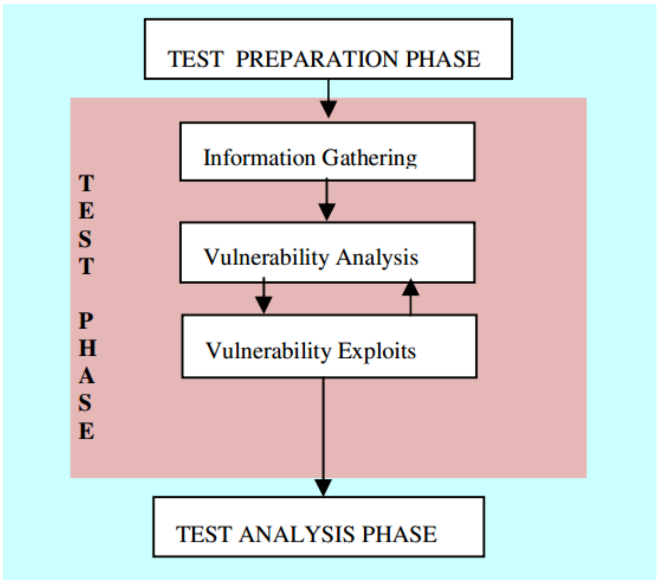


Figure 6. Penetration Test Phases [18].

Despite code review was not confirmed with penetration test, the common flaws shown in Table 1 are enough to demonstrate that touch points must be considered in software development lifecycle. A hacker or an attacker with moderated knowledge can exploit these software flaws easily.

VI. CONCLUSION AND FUTURE WORK

This assessment showed the importance of observing the security aspects in the software development lifecycle. The standards used for regulation of medical device software do not take into account security concerns. These aspects can make all difference in final software security and also in patient safety.

It is responsibility of *QSRs* deal with security concerns clearly. In general, standards for normalization of validations and verification are worried about functional aspects of software operation. Security issues are generally collateral problems that persist in all phases of software lifecycle, until software finishes its production life.

The monitor software used in the analysis was not designed, and as consequence, built with security concerns. So, every kind of security issue can appear in assessment. Since assessed software is in earlier stage of development, it is easier to map problems, flaws, issues and vulnerabilities and create a plan to mitigate them.

Generally, this kind of assessment produces lots of confidential results, and it is difficult to show them without brake non-disclosure agreements and/or reveal sensitive information about software internal structure. More relevant results were discussed directly with design and implementations teams involved in research project.

Unfortunately, securities problems are only solved when entire team involved in software construction are conscious about how it can affect in software operation.

To create this kind of conscience lots of actions are important. But, only organizations that have a security culture and security personal with secure coding and assessment skills can address these actions correctly.

In next steps, a complete penetration test will be performed, trying to exploit vulnerabilities found in code reviews and confirming that risks mapped were mitigated.

ACKNOWLEDGMENT

The authors would like to thank Software Engineer Group from Computer Science Department at University of Brasilia for allowing the analysis performed at Monitor software. This software is part of a research project entitled *Ambient Assisted Living Product Lines.*

REFERENCES

[1] H. Fraser, Y. Kwon, and M. Neuer, The future of connected health devices, IBM Institute for Business Value, New York, 2011.

[2] IEC 62304, Medical device software – Software life cycle processes, 1st ed, Geneva, 2006.

[3] ISO 27799, Health informatics – Information security management in health using ISO/IEC 27002, 1st ed, Geneva, 2008.

[4] J. Radcliffe, "Hacking Medical Devices for Fun and Insulin: Breaking the Human SCADA System", Black Hat Conference, Las Vegas, 2011, http://media.blackhat.com/bh-us-11/Radcliffe/BH_US_11_Radcliffe_Hacking_Medical_Devices_WP.pdf, 01.01.2013

[5] D. Halperin, et al., “Pacemakers and Implantable Cardiac Defibrillators: Software Radio Attacks and Zero-Power Defenses”, Proceedings of the IEEE Symposium on Security and Privacy, Oakland, 2008, pp. 129-142, doi:10.1109/SP.2008.31.

[6] S. R. Rakitin, “Coping with Defective Software in Medical Devices”, Computer Magazine - IEEE Computer Society, v. 39, 2006, n. 4, pp. 40-45, doi: 10.1109/MC.2006.123.

[7] D. A. Vogel, Medical Device Software Verification, Validation, and Compliance, Boston: Artech House, 2011.

[8] G. McGraw, Software security: building security in, Boston: Addison Wesley Professional, 2006.

[9] U.S. Food and Drug Administration, Guidance for the Content of Premarket Submissions for Software Contained in Medical Devices. 1st ed, New Hampshire, FDA, 2005.

[10] Open Web Application Security Project, OWASP Top Ten – 2010 The Ten Most Critical Web Application Security Risks, CC:OWASP, 2010.

[11] K. Hall, “Developing Medical Device Software to IEC 62304”, European Medical Device Technology Magazine, v. 1, n. 6, June 2010.

[12] R. J. Anderson, Security Engineering: A Guide to Building Dependable Distributed Systems, 2nd ed, Indianapolis: Wiley Publishing inc, 2008.

[13] J. Six, Application Security for the Android Platform, 1st ed California: O’Reilly Media, Inc, 2012.

[14] F. Long, D. Mohindra, R. C. Seacord, D. F. Sutherland, and D. Svoboda, The CERT Oracle Secure Coding Standard for Java, 1st ed, Michigan: Pearson Education Inc, 2012.

[15] D. Stuttard and M. Pinto, The Web Application Hacker’s Handbook: Finding and Exploiting Security Flaws. 2nd ed Indianapolis: Wiley Publishing inc, 2011.

[16] M. Paul, Official (ISC)2 Guide to the CSSLP, 1st ed, Florida: CRC Press, 2011.

[17] V. Nunes, P. Fernandes, V. Alves, and G. Rodrigues, “Variability Management of Reliability Models in Software Product Lines: an Expressiveness and Scalability Analysis”, I: SBCARS - Simpósio Brasileiro de Componentes, Arquitetura e Reutilização de Software, Natal - Brazil, 2012, pp. 113 - 122.

[18] A. G. Bacudio, X. Yuan, B. B. Chu, and M. Jones , “An Overview of Penetration Testing”, International Journal of Network Security & Its Applications (IJNSA), v.3, 2011, n.6, pp. 19-38, doi: 10.5121/ijnsa.2011.3602.