Security Quality Assurance on Web Applications

Rodrigo Elia Assad[1,2], Felipe Ferraz[1,2], Henrique Arcoverde[3,] Silvio Romero Lemos Meira [1,2]
[1]Centro de Estudos e Sistemas Avançados do Recife(CESAR) - Recife – PE – Brazil
[2]Centro de Informática
Universidade Federal de Pernambuco (UFPE) – Recife, PE – Brazil
[3]Tempest Security Inteligence
assad@cesar.org.br, fsf3@cin.ufpe.br, henrique@tempest.com.br, srlm@cesar.org.br

Abstract: Historically, it is well known that issues related to security of software applications are normally omitted by the development teams owing to a lack of expertise or knowledge in security policies. With the emergence of WEB technologies, this situation became more serious. Entire systems, complex or not, have outstanding access availability and therefore are highly vulnerable to threats. This work aims to discuss how the security requirement, design patterns and tests should be elaborated in order to making easier the execution of its tests and consequently improving the quality of the solution developed.

Keywords-security requirements; design patterns; security tests validation, quality assurance

I. INTRODUCTION

Since the popularization of the Internet through its commercial use occurred during the decade of 1990, attacks on computer systems have become more frequent. Initially the attacks was more focused on operating systems and network services, as can be seen in the attacks reports generated from various institutes such as CERT [47].

With the rapid growth of attacks, companies, governments, universities invested heavily in security solutions, such as firewall, intrusion detection systems, anti-virus, patch management and so on. Also there was investment on development of security procedures and processes for managing information, such as ITIL, COBIT and SOX [46]. And also a definition of specific legislation to support the security analysts.

All these initiatives, associated with maturation time, related to security issues comprehension and security standards adoption, occurred from 1997 to 2007; the rate of attacks reported to security holes in operating systems and computer networks have decrease significantly, as shown in Figure 1.

Analyzing these numbers, we see that the definition of procedures, comprehension of security flows produces an improvement on a perceived security quality - QA - Quality Assurance - in relation to services provided by system administrators, security consultants and security engineers that support computer networks and operation system.

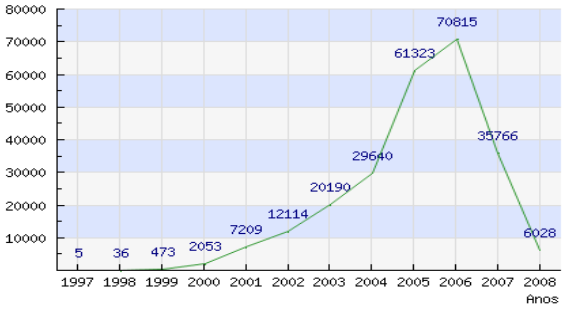

Figure 1: Attacks on network operating systems and reported by CERT.br [47]

The improvement and maturation of Security Quality Assurance procedures do network and operation systems resulted in change of security focus, now applications have become the primary target.

It is undeniable that the security problems still persist, however, are not only related to flaws in operating systems or network services, but the major focus has changed and is currently in web applications, as seen in Figure 2.

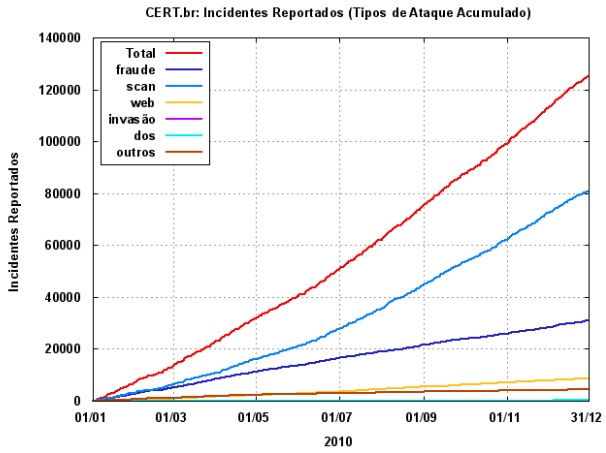

Figure 2: Attacks reported on 2010 to Cert.br [47]

The attacks on web applications and began more popular on 2007. It´s can be evidenced in several ways, among them, through consultations on Google cache

 ISBN: 978-1-61208-165-6

showed on Figure 4. As observed before 2007 there are few records of consultations about web application security. Using the same methodology shown in Figure 3 queries related to network security - a subject of greater scope related to operating system security and network services - has been decreasing year after year as a result of the quality assurance process described before.

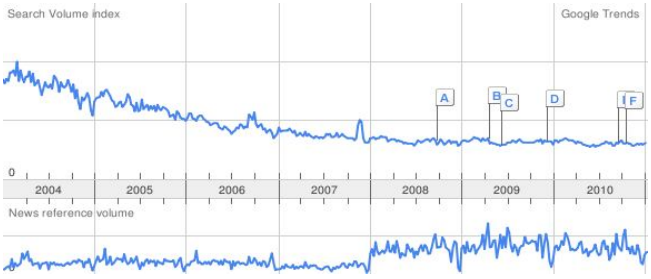

Figure 3: Query to Google on “network security"[50]

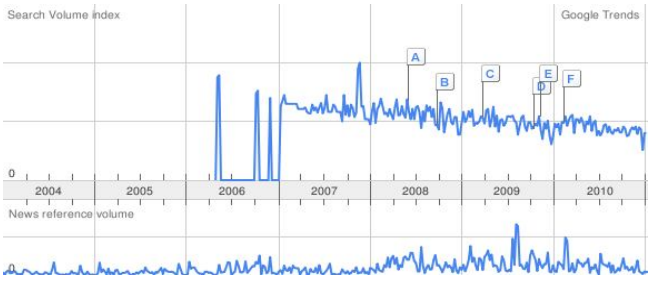

Figure 4: Query to Google on “web application security "[50]

Another fact should be observed that’s collaborate to attacks migration to web applications, we have in this period the emergence of applications WEB2.0, Web3.0 [37], consolidation of the browser as a gateway to all applications, the emergence of API's development as the proposed Google apps and Microsoft Live and more recently the cloud computing [38][44][48].

We can believe that, as happened as with the attacks on operating systems and network services, to protect web application we will require efforts on research development, new product development and procedures specifications, and consequently the maturation of software developers in order to improve the code produced for web applications, it can be called SSQA: Security Software Quality Assurance:.

To define the security Quality Assurance demanded by an application, it is required experienced and trained stakeholders with abilities in all disciplines with a focus on security. Throughout the development of a web application, it is important that the activities of elicitation and specification of requirements to be followed by architecture definition and a process of validation. It is essential to track and approve if the security requirements are being satisfied on applications. The traceability of functional and non-functional security requirements is naturally a complex task, because in general, they affect the entire system. To carry out successful safety tests on a application, is necessary to identify what types of vulnerabilities could enable attacks on the system. In other words, we must understand where the flaws may be present to learn how to avoid them.

From this point of view, we identified that one of the possible causes to security flaws relies on the low quality of software security requirements and consequently in its implementation, validation and tests phases.

It should consider the present scenario of IT companies in relation to technologies used in the development of web applications, we have the main highlights:

a) Use agile methodologies

b) Software reuse

c) Development framework

This section presents a proposal for the integration of the above themes, throws specifying a security quality assurance process that can be used by companies to promote the development of secure applications on certain assumptions, keeping the agreed deadlines and focusing on quality assurance of the safety of software. It´s examines the possibility of adopting the same methodology used successfully between 1997 and 2007 that brought a significant drop in network security problems, they are:

a) Understanding of attacks and its operating mechanism

b) Development of defense models in relation to existing technology

c) Adopting an agile and reusable

d) Establishment of a pricing mechanism for the easy development of secure solutions.

II. SECURITY QUALITY ASSURANCE WEB

The proposal of this paper is to guarantee the security quality assurance of web applications, by defining a methodology that could be reused and agile. So the first objective is identifying the main problems of web application. To do it, we used a real case scenario of a security company of Brazil called Tempest Security Intelligence [49] that sales web penetration test service.

The whole universe of the research described here corresponds to 467 reported vulnerabilities in the Tempest Security Intelligence analysis projects and web application ethical hacking of web applications, not considering the analysis projects of infrastructure. An importantly point is the vulnerabilities are spread across various customers and

 ISBN: 978-1-61208-165-6

do not correspond to points of vulnerability to be explored but real flows on web application. For example, given an analysis in the foo app, there are 15 points where you can perform SQL attacks, however, the vulnerability is reported only once. The numbers represent only the vulnerability in the application and not the amount of exploitable points in each application.

The research uses as base the perspective of the OWASP Top 10 2010 [45] version vulnerabilities, successor version of OWASP Top 10 2007 version, so the data used are restricted to projects reported between the years 2008 and 2010. On Figure 5 we present the workflow used.

Data collection corresponds to real cases but to preserve the client we uses a fictitious names.

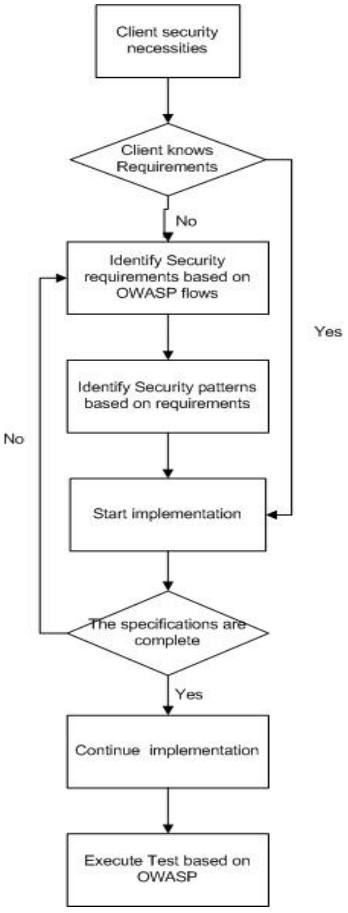

Figure 5: Software Security Quality Assurance workflow

The sample profile collected is determined in two characteristics: year of publication and type of vulnerability. First characteristic determines the year in which the vulnerability was discovered and published to the client. As previously described the data for the years 2008, 2009 and 2010. It was observed that 17% of vulnerabilities were reported in 2008 (Figure 6), 28% were reported in 2009 and 55% were reported in 2010. (Figure 6)

Figure 6: Vulnerabilities per year [49]

Another information collected was the type of collected vulnerable applications, showed on Figure 6.

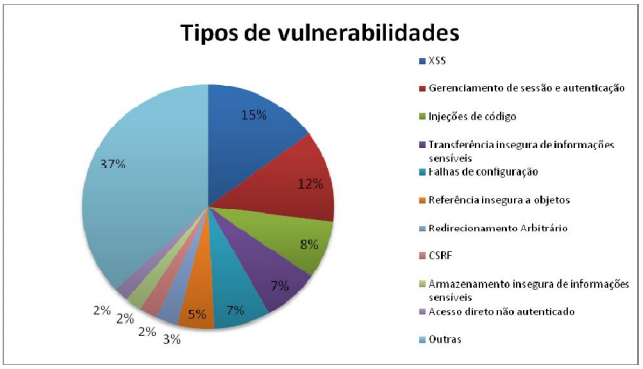

Figure: 7 Types of vulnerabilities [49]

Resuming, it is possible to observe that 10 of vulnerabilities reported by Tempest Security Intelligence in the years 2008, 2009 and 2010 are: 15% of XSS vulnerabilities are observed, session management and access control are 12% of the vulnerabilities, 8% of the vulnerabilities are code injection, 7% of the vulnerabilities are flaws in the configuration, with 7% of the vulnerabilities are flaws transfer insecure credentials, the reference objects unsafe to correspond to 5% of the vulnerabilities, 3% of vulnerabilities are related to arbitrary redirection, 2% of vulnerabilities are related to direct access to unauthenticated, 2% for safe storage of sensitive and 2% of CSRF. The table bellow makes a comparison between Owasp reports and Tempest results.

TABLE 1: COMMON VULNERABILITY ACCORDING TEMPEST.

Tempest Top 10			
2008	**2009**	**2010**	**General**
XSS	XSS	Session authentication management	XSS
Code injection	Session authentication management	XSS	Session authentication management
Unsecure	Code injection	Configuration	Code injection

transmission of sensitive data		flows	
Session authentication management	Unsecure transmission of sensitive data	Unsecure reference to objects	Configuration flows
Configuration flows	Configuration flows	Code injection	Unsecure transmission of sensitive data
Direct access with no authentication	Unsecure reference to objects	Unsecure transmission of sensitive data	Unsecure reference to objects
Unsecure reference to objects	Direct access with no authentication	Unsecure sensitive and storage information	Arbitrary redirect
Unsecure sensitive and storage information	Arbitrary redirect	CSRF	CSRF
CSRF	CSRF	Arbitrary redirect	Unsecure sensitive and storage information
Arbitrary redirect	Unsecure sensitive and storage information	Direct access with no authentication	Direct access with no authentication

III. REQUIREMENTS

The requirements engineering on a business, systems, applications and components is more than just document that describes functional requirements of the application. Even though, most system analysts dedicate the bigger art of their time to elicit some quality requirements such as interoperability, availability, performance, portability and usability, many of them still sin with regard to addressing issues related to security.

Unfortunately, documenting specific security requirements is difficult. These tend to cause a high impact for many functional requirements. Furthermore, security requirements are usually expressed in a document the terms of how to achieve security not as the problem that needs to be resolved [27].

Most system requirements analysts have no knowledge in Security, the few who received some training had only a general overview of some security mechanisms such as passwords and encryption rather than meet real requirements in this area [5][28][29].

Security requirements deal with how the assets of a system must be protected against any kind of evil [27][30]. An asset is something within the system, tangible or not, that must be protected [31]. A threat or harm from which a system must be protected, is a potential vulnerability that can reach a well. A vulnerability is a weakness of a system that tends to exploit an attack. Security requirements are constraints on the functional requirements in order to reduce the scope of vulnerabilities [27].

On the Bellow table we make a comparison between Donald Firesmith [28][29] security requirements proposal: Identification, Authentication, Authorization, Non-Repudiation, Privacy, Immunity, Integrity, Intrusion Detection, Security Audit, Maintenance Systems Security and Physical Protection; and OWASP Web vulnerability list, it relates each vulnerability and the correspondent requirements

TABLE 2: REQUIREMENTS X VULNERABILITIES

Requirement	Owasp Test
Identification	OWASP-IG-003, OWASP-IG-004
Authentication	OWASP-AT-001, OWASP-AT-002, OWASP-AT-003, OWASP-AT-004, OWASP-AT-005, OWASP-AT-006, OWASP-AT-007, OWASP-AT-008, OWASP-AT-009, OWASP-AT-0010
Authorization	OWASP-AZ-001, OWASP-AZ-002, OWASP-AZ-003
Imunity	OWASP-IG-005, OWASP-IG-006, OWASP-CM-002, OWASP-CM-003, OWASP-CM-006, OWASP-CM-008, OWASP-DV-001, OWASP-DV-002, OWASP-DV-003, OWASP-DV-004, OWASP-DV-005, OWASP-DV-006, OWASP-DV-007, OWASP-DV-008, OWASP-DV-009, OWASP-DV-0010, OWASP-DV-0011,OWASP-DV-0012,OWASP-DV-0013,OWASP-DV-0014, OWASP-DV-0015,OWASP-WS-002, OWASP-WS-003, OWASP-WS-004, OWASP-WS-005, OWASP-WS-006, OWASP-WS-007, OWASP-AJ-002
Integrity	OWASP-SM-001, OWASP-SM-002, OWASP-SM-003, OWASP-SM-004, OWASP-SM-005
Intrusion Detection	OWASP-CM-005
Non – Repudiation	
Privacity	OWASP-IG-001, OWASP-CM-001
Security Audity	
Fault Tolerance	OWASP-DS-001, OWASP-DS-002, OWASP-DS-003, OWASP-DS-004, OWASP-DS-005, OWASP-DS-006, OWASP-DS-007, OWASP-DS-008
Physical protection	
Maintenance of Security Systems	OWASP-CM-004, OWASP-CM-007

Since we have a relation that puts security vulnerability and system requirements we can elaborate reusable system

requirements based on security flows that could be addressed by system analysts during the project conception.

IV. DESIGN PATTERNS

Now, that we have a relation between security vulnerabilities and system requirements the next step is try to use a design patterns concept to propose a methodology to use it, giving to the user a choice to make a latter implementation of security code. To do it we use a classification given by the GoF Design Patterns.

Initially, we will consider the following requirements: Identification, Authentication, Authorization, Non Repudiation and Privacy. Reviewing these requirements, we can observe that all of this belongs to the same subject, identification of an actor, be a user, system or other entity that interacts with the system in question. All of them deal with the identification of an actor. Respectively have the actor ID, proof of ID, permissions of identity, confirmation of the shares of the entity and finally the secrets or secret identity. All of these requirements revolve around the creation of an identity.

Going forward on requirements analysis, we can separate the requirements for immunity and integrity, as two conditions that directly affect the structure of the system. From this viewpoint, the requirements for immunity vision ensure that the system is immune to contamination by parts of the actors and the requirements of Integrity vision ensure that the structure of an integrated system communication between these actors. Both requirements have a direct relation with the structure of the system since it will be necessary to change the structure of the system to adopt solutions to these requirements.

Following, we have the requirements for Intrusion Detection, Security Audit and Fault Tolerance, which deal with issues related to actions taken by the system. In the first case detection, we have a requirement that works as a prevention, which aimed to provide a mechanism for detection and notification in case of unauthorized access, since the audit comes as a mechanism to work issues in a more reactive, or attitudes that can be taken from the evidence and observation of actions, an audit requirement must include the registration of shares as well as mechanisms for future reference [11], different fault tolerance as well as reactive, which defines the behavior of the system will have in case of failure, is also to ensure that preventive flaws in system entities do not jeopardize the rest of the system. Therefore, the requirements of work on the issue of the conduct taken within the system.

Finally analyzing the requirements for Maintenance of System Security and Physical Protection have, this is a requirement that is more than physical matter, as the name refers, where the concern goes beyond the scope of software, both outside the scope of our analysis. Since the requirement for maintenance has a horizontal behavior in relation to other requirements, since this deals with the maintenance of the system's other needs related to security, he is indirectly responsible for such requirements needs. Appears not a requirement for so considerable in terms of software and one that is the sum of the other requirements.

Organizing them according to their characteristics are:

1) Requirements Identification, Authentication, Authorization, Non-Repudiation and Privacy and related creation.

2) Integrity and Immunity Requirements related to the structure of the system.

3) Requirements for Intrusion Detection, Audit and Fault Tolerance-related behaviors of the actors in the system.

4) Requirements for Maintenance of Security Systems related to the other requirements.

Under this approach, using some of the classifications of GoF, we can separate the requirements according to their purposes. From the characteristics presented, we will separate them into three groups according to this criterion purposes, they are, Creation, Structural and Behavioral.

TABLE 3: RELATING PATTERNS AND REQUIREMENTS

Purpose		
Creation	**structural**	**behavioral**
Identification requirement	Immunity Requirement	Intrusion detection Requirement
Authentication Requirement	Integrity Requirement	Security audit Requirement
Authorization Requirement		Fault tolerant Requirement
Non-repudiation Requirement		
Security Maintenance Requirement		

Physical protection requirements that deal with physical issues related to the physical system are not addressed within this framework.

V. CASE STUDY

A) Reusable requirements

The tasks described by this article were used on the development of some IT projects on C.E.S.A.R (Center of Studies and Advanced Systems of Recife) and UNIMIX. These IT companies are needing to realize a detailed analysis of security issues in some projects with the purpose of making sure that system that are considered critical be tested and validated. As a consequence, this ensures that everything agreed on the contract is respected

 ISBN: 978-1-61208-165-6

by the service provider and cannot be questioned by the client.

Due to contracts issues, we are not allowed to give any further information about the context in question. However, some points must be cited, such as:

a) The process of writing requirements has been validated in three projects with companies that act on these sectors: telecommunications and informatics. In these cases, the objective is only write the requirements document and the proposed methodology was employed. As a result, customers noticed an improvement in the problems understanding in early stages of the project.

b) During the risk analysis, the suggested changes in the templates for requirements elicitation indicated a greater understanding of the problems, possible solutions for them and mitigating strategies.

c) Preparation of a repository of reusable security requirements and their test cases based on the recommendations of tests developed by OWASP. This was the case with the requirements of P2P FINEP project, which aims to deploy solutions in environments peer-to-peer. Their requirements and test cases were used for decision making in relation to which requirements and test cases as well as risk analysis for the management solution descktop dreams (HTTP: / / www.dreamsweb.com. br)

d) We have a case study in the development of corporative site of a technology company of Pernambuco. This scenario was run throughout the full proposed cycle in this paper. It was observed that: a) there was significant improvement of the safety requirements of the portal, b) in the testing phase were found about 11 flaws in the site that did not meet the requirements, some of them quite serious, c) Another project in onset may benefit from the basic set of requirements, d) Part of the scripts could also be reused. Unfortunately for security reasons the company was not authorized to divulge more details of the results, as problems are identified security.

e) Observers that the methodology described here is used with extreme efficiency and trends in the proposals brought to the development of systems that must function in an environment of cloud computing. This is because in this environment issues of SaaS, PaaS and IaaS introduce the characteristics of infrastructure as something bringing programmable horizontal scalability for applications. It is undeniable that as we have the scalability of an application being made across the board problems and new security risks arise. These problems not previously considered relevant. Mainly on issues related to security [44]. However, the proposed solutions have a way to specify and reuse them efficiently because the strategies do not vary much scalability. The main result of this work, we observed an improved understanding of the technical requirements and their implementation by software security engineers and the ability to produce more accurate tests and that met the needs of customers. Thus reducing the need for correction of deficiencies identified in the test phase, which is one of the main mistakes made in building secure software [43].

Also as a result it will have a better quality software, on the point of view of ensuring the functionality specified by carrying out a process of validation and elaboration.

Another important result presented in this paper is to provide project managers the ability to quantify risk in relation to the implementation or not a particular requirement before starting the coding phase.

As a consequence of this work, we observed a satisfactory improvement on the comprehension of technical needs and its implementation by software engineers besides the ability to produce test more precise that meet clients need. Consequently, we were able to develop software with better quality from the point of view of the functionality assurance through the performance of a validation process more elaborated.

B) Design Patterns and a late implementation

The purpose of the case study was to validate the proposed relationship between the GoF design patterns as a way to represent security requirements. As mentioned in the work we try to have a more practical assessment of our study to evaluate the feasibility of using these standards as a tool in implementing security requirements and to facilitate the understanding of security requirements for developers in genera during the software development process.

Initially our study was conducted in a project expected to last 3 (three) months, we will call this a Test System. This project would serve initially as a proof of concept for a larger project, with issues related to client confidentiality and NDA cannot go into further detail concerning the applicant, project name and other sensitive information to be omitted.

In a second moment relationships proposed in this paper was applied again in the second Test System that time this system was already in a more consolidated stage requiring greater attention as we shall see below.

Finally, a third opportunity was presented to us where we suggest an approach to two related structures created in this work. Unlike the two previous occasions the third opportunity is still being implemented.

 ISBN: 978-1-61208-165-6

For these tree applications at the beginning of the project the security requirements was not so clear, but once completed the implementations we could see a positive result of implementing the proposals made by the job. One of the best insights that should be observed was that the changes on requirements were not very intrusive, low impact and easy modification.

So the opportunity to apply the propositions made in this initial work in a context outside of the web, mainly located on the server side, served as an initial validation of a positive result. Besides these the possibility of making a second application using a macro context of the application, addressing GWT, RPC calls, proved satisfactory.

Furthermore the application of the propositions in a macro context has generated the perception that the adoption of standards, as related to ambiguous or poorly written requirements, may present as a data point requiring a second more detailed approach to understanding how to address . Still, the approach was extremely valid and consistent highlighting the importance of future studies mentioned in the next section.

VI. CONCLUSION

The software quality cannot be measured only by the assurance of the execution of a process, but by the results of its execution and necessary validations. Within this context, this paper aimed to define tasks, recommendations and process that should be introduced on the cycle of software development with the purpose of guiding the test and validation phase to produce more elaborated and precise results from the point of view of security issues.

The process proposed by this paper is being introduced on the software cycle development s at C.E.S.A.R as specific security needs are required.

The adoption of this process allowed making a more critical analysis of the new features introduction on new projects as well as the test team comprehension at executing these tasks thus improving the software quality observed by the clients.

As a final contribution, we were able to validate the proposal software security requirements reuse and its test cases in other projects inside C.E.S.A.R, proving that this process is extensible as proposed.

ACKNOWLEDGMENT

This work was partially supported by the National Institute of Science and Technology for Software Engineering (INES http://www.ines.org.br), funded by CNPq and FACEPE, grants 573964/2008-4 and APQ-1037-1.03/08.

REFERENCES

[1] R. Lutz, "Software engineering for safety: a roadmap," Proceedings of the Conference on The Future of Software Engineering, ACM, 2000, pp. 213–226.

[2] B. Matt, "What Is Computer Security?," Computer, 2003, pp. 67-69.

[3] I. Sommerville, T. Rodden, P. Sawyer, R. Bentley, and M. Twidale, Integrating Ethnography Into the Requirements Engineering Process, 1993.

[4] D.G. Rosado, C. Gutiérrez, E. Fernández-medina, M. Piattini, P.D. Universidad, C. Real, D. Grosado, E. Fdez-medina, S.T. Calle, and M. Tovar, "A Study of Security Architectural Patterns," Information Systems, vol. 1, 2006, pp. 2-9.

[5] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security," Urbana, vol. 51, p. 61801.

[6] P.T. Devanbu and S. Stubblebine, "Software Engineering for Security: a Roadmap," The future of Software Engineering, ACM Press, 2000, pp. 227-239.

[7] N. Yoshioka, H. Washizaki, and K. Maruyama, "A survey on security patterns," Progress in Informatics, 2008, p. 35.

[8] G. Sindre and A.L. Opdahl, "Eliciting security requirements with misuse cases," Requirements Engineering, vol. 10, 2004, pp. 34-44.

[9] W.C. Summers, "Password Policy: The Good, The Bad, and The Ugly.", Proceedings of the Winter International Symposium on Information and Communication Technologies, 2004

[10] P. Samarati and S.D. di Vimercati, "Access Control: Policies, Models, and Mechanisms," FOSAD, R. Focardi and R. Gorrieri, Springer, 2000, pp. 137-196.

[11] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns : Integrating Security and Systems Engineering (Wiley Software Patterns Series), John Wiley & Sons, 2006.

[12] F. Khomh and Y.G. Gueheneuc, "Do Design Patterns Impact Software Quality Positively?," Software Maintenance and Reengineering, 2008. CSMR 2008. 12th European Conference on, 2008, pp. 274-278.

[13] J. Katz and Y. Lindell, Introduction to Modern Cryptography (Chapman & Hall/Crc Cryptography and Network Security Series), Chapman & Hall/CRC, 2007.

[14] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, Design Patterns: Elements of Reusable Object-Oriented, Addison-Wesley Professional.

[15] I. 7498-2, Information processing systems -- Open Systems Interconnection -- Basic Reference Model -- Part 2: Security Architecture, 1989.

[16] D. Zerkle and K. Levitt, "NetKuang -- A Multi-Host Configuration Vulnerability Checker," in Proceedings of the 6th USENIX Unix Security Symposium, 1996.

[17] P. Steiner, "On the internet nobody knows you´re a Dog," The New Yorker, vol. 69, 1993, p. 61.

 ISBN: 978-1-61208-165-6

[18] P. Kunyu, "An identity authentication system based," Identity, 2009.

[19] Z. Wang, M. Li, M. Chen, and C. Chang, "A New Intelligent Authorization Agent Model in Grid," 2009 Ninth International Conference on Hybrid Intelligent Systems, 2009, pp. 394-398.

[20] G. Dhillon, Principles of Information Systems Security: Texts and Cases, Wiley, Ed. Virginia Commonwealth Univ March 2006

[21] H. Peiris, L. Soysa, and R. Palliyaguru, "Non-Repudiation Framework for E-Government Applications," 2008 4th International Conference on Information and Automation for Sustainability, 2008, pp. 307-313.

[22] J. Adikari, "Efficient Non-Repudiation for Techno-Information Environment," First International Conference on Industrial and Information Systems, 2006, pp. 454-458.

[23] L.F. Soares, G. Lemos, and S. Colcher, Redes de Computadores: das LANs, MANs e WANs às redes ATM, Rio de Janeiro: Campus, 1995.

[24] C. Alexander, S. Ishikawa, and M. Silverstein, A pattern Language, Oxford University Press.

[25] R.P. Gabriel, Patterns of software: tales from the software community, Oxford University Press, Inc. New York, NY, USA, 1996.

[26] I. Oliveira, "Uma Análise de Padrões de Projeto para o Desenvolvimento de Software Baseado em Agentes," 2001.

[27] F. Khomh, Y. Guéhéneuc, and P. Team, "An empirical study of design patterns and software quality," 2008, pp. 1-19.

[28] P.S. Sandhu, P.P. Singh, and A.K. Verma, "Evaluating Quality of Software Systems by Design Patterns Detection," 2008 International Conference on Advanced Computer Theory and Engineering, 2008, pp. 3-7.

[29] M. Bernardi and G. Di Lucca, "Improving Design Pattern Quality Using Aspect Orientation," 13th IEEE International Workshop on Software Technology and Engineering Practice, 2005, Ieee, 2005, p. 206–218.

[30] C.B. Haley, R.C. Laney, and B. Nuseibeh, "Deriving security requirements from crosscutting threat descriptions," AOSD '04: Proceedings of the 3rd international conference on Aspect-oriented software development, New York, NY, USA: ACM Press, 2004, pp. 112-121.

[31] D. Firesmith, "Engineering security requirements," Journal of Object Technology, vol. 2, 2003, p. 53–68.

[32] D. Firesmith, "Analyzing and Specifying Reusable Security Requirements," Eleventh International IEEE Conference on Requirements Engineering (RE'2003) Requirements for High-Availability Systems (RHAS'03) Workshop, Citeseer, .

[33] C.B. Haley, J.D. Moffett, R. Laney, and B. Nuseibeh, "A framework for security requirements engineering," SESS '06: Proceedings of the 2006 international workshop on Software engineering for secure systems, New York, NY, USA: ACM Press, 2006, pp. 35-42.

[34] nformation Technology - Security Techniques - Evaluation Criteria for IT Security, Geneva Switzerland: ISO/IEC Information Technology Task Force (ITTF) .

[35] M. Weiss and H. Mouratidis, "Selecting Security Patterns that Fulfill Security Requirements," 2008 16th IEEE International Requirements Engineering Conference, 2008, pp. 169-172.

[36] Structural Patterns at Source Making.", http://sourcemaking.com/structural_patterns, last accessed 8/10/2011 .

[37] G. Inc, "GWT, Google Web Toolkit," http://code.google.com/webtoolkit/, last accessed 8/10/2011.

[38] J. Company, "Hibernate," http://www.hibernate.org/, last accessed 8/10/2011.

[39] Gilead, Generic Light Entity Adapter, http://noon.gilead.free.fr/gilead/, last accessed 8/10/2011.

[40] P. Pawlak, B. Sakowicz, P. Mazur, and A. Napieralski, "Social Network Application based on Google Web," Source, 2009, pp. 461-464.

[41] M. Dhawan and V. Ganapathy, "Analyzing Information Flow in JavaScript-Based Browser Extensions," 2009 Annual Computer Security Applications Conference, 2009, pp. 382-391.

[42] E. Ofuonye and J. Miller, "Resolving JavaScript Vulnerabilities in the Browser Runtime," 2008 19th International Symposium on Software Reliability Engineering (ISSRE), 2008, pp. 57-66.

[43] Meier J. Web application security engineering. IEEE Security & Privacy Magazine. 2006;4(4):16-24.Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnuber=1667998, last accessed 8/10/2011.

[44] SUN, Introduction to Cloud Computing architecture White Paper 1st Edition, June 2009

[45] Owasp, 2008, owasp testing guide 2008 v3.0. Disponível em: http://www.owasp.org/index.php/category:owasp_testing_project, last access: 06/05/2009

[46] PMBOK, Project Managment Institute (PMI) standards committee: A guide to the Project Management Body of Knowledge (PMBOK) Third edition,2008.

[47] Us-cert - technical cyber security alerts, 2009. Disponível em: http://www.us-cert.gov/cas/techalerts/ . Last access: 29/04/2009

[48] Cloud Computing Use Case Discussion Group Version ; Cloud Computing Use Cases A white paper produced by the 2.0 30 October 2009

[49] Tempest Security Intelligence, www.tempest.com.br, last accessed 05/05/2011

[50] Google Trends Service, www.google.com/.trends, last accessed 8/10/2011.

 ISBN: 978-1-61208-165-6