Usability Support Security Patterns

Susana Romaniz[1], Marta Castellaro[1], Juan Carlos Ramos[1], Ignacio Ramos[1]

[1] Facultad Regional Santa Fe - Universidad Tecnológica Nacional,
Lavaise 610 (S3004EWB) Santa Fe Argentina

{sromaniz, mcastell, jramos, iramos}@frsf.utn.edu.ar

Abstract. The main feature of secure software lies in the nature of processes and practices used to specify, design, develop and implement software. Security patterns applied the concept of pattern in the security realm. Its description helps to capture immediately the essence: what is the problem to which attends and what the proposed solution is. The different formats that exist for its description and the multiplicity of sources make its discovery demand effort that discourages the systematic use by potential recipients. This paper presents the prototype of a catalogue that seeks to establish a bridge between the knowledge and experience security experts and the needs of knowledge of software development teams.

Keywords: Security patterns. Security intelligence. Security patterns catalogue.

1 Security in the software development process

The main feature of secure software lies in the nature of the processes and practices used to specify, design, develop and deploy the software [1]. A project that adopts an improved security software development process incorporates a set of practices that reduce the number of exploitable flaws and bugs. Over time, these practices become more systematic, so it should decrease the likelihood that such vulnerabilities are present in the software at the moment that releases. The results in the field of research and experiences in the industry indicate the importance of reducing such potential vulnerabilities as early as possible within the software development lifecycle. The adoption of improved security processes and practices is much more profitable than the solution widespread today to develop and release patches for the operating software [2].

This early attention of the security has to do with the adoption of a set of activities that make possible the security integration in the software development lifecycle [3], including [4]: 1) identify security objectives, 2) apply security design guidelines, 3) create threat models, 4) conduct security architecture and design reviews, 5) complete implementation security reviews, and 6) run deployment security reviews.

We note the active development of tools and methods for testing that allows assessing the robustness and resilience of software products and their underlying infrastructure under attack conditions (for example, those used in penetration testing). Namely, there is intelligence domain of attacks that exploit vulnerabilities and compromise the security of software-intensive systems. All this produces a complex deal scenario.

There are criteria that help to secure software production [5]:

- Keep in mind that the security and cost of production of a software system depends strongly on the knowledge about its requirements [6].
- Include security treatment in each of the different stages of the software development cycle is an accepted criterion for improving the security of the final product. [7]
- Incorporate security patterns, which represent the best practices achieved by the industry in order to stop or limit security attacks [8].

In the secure software development process' case, security patterns are a way to bridge the gap between theory and practice. Although there are theoretical approaches, they are limited to relatively complex systems, and require a grade of knowledge and experience that is not available at the necessary level.

Then, we may infer that promote the use of security patterns to guide and drive the building of secure software development models, from the early stages of the process, is an approach that will allow us to ensure greater security in the behavior of a software product.

2 Security Patterns

The concept of "pattern" is known by the community as a solution to common problems in software development, whose effectiveness has been verified by solving similar problems in the past and that is reusable (applicable to different design problems in various circumstances). In the case of the security software aspects, which are found throughout all phases of development, its original definition [9] states that: "*Each pattern is a three-part rule, expressed as a relation between a certain context, a certain system of forces that occur repeatedly in this context, and a certain software settings that allows these forces to resolve themselves.*"

Extending the concept to software security, security patterns documented well known solutions to recurring problems of information security, allowing an efficient transfer of experience and knowledge. They apply the pattern concept to the realm of security, describing a particular recurrent security problem occurring in a specific context and presenting a well-proven generic solution accepted by the community of experts [10]. To make explicit the assumptions under which apply their solutions are applicable, reduce the risk of inappropriate use.

The solution proposed by a security pattern consists of a set of interacting roles that can be organized in multiple structures (applicable to the phases of requirements analysis, design, coding, testing or implementation, as appropriate pattern) concrete, as well as a process to create a particular structure in these [8]. According to what is expressed in [11] "a pattern defines a process and a thing: the 'thing' is created by the 'process'." Security patterns can be categorized according to a point of view associated with a software development lifecycle [12]. In this way there are patterns to the requirement phase, patterns to the design phase and patterns for the implementation phase. In general terms, guide the analysis through the context of security patterns allows integrate the problem addressed and the forces present that determine the problem.

Specifically, highlights the following advantages in the use of an approach to the use of patterns in security treatment: (i) express the basics security in a structured and understandable way; (ii) its representation is familiar to software developers and system

engineers, a key part of their audience; (iii) the patterns already are used to capture the knowledge about the system and the organization, then using the patterns to capture security knowledge help to improve security in the systems lifecycles, where results clearly needed. In particular regard to its development, in the last years different security groups have been specifying different patterns, as well as have made efforts for classification purposes [13, 14].

2.1 Security patterns description

Often refers to the model POSA (*Pattern-Oriented Software Architecture*) model developed by Buchman and others [15] to describe the context and the use of security patterns. But you can also see the existence of different descriptive models [16]; even, in many cases they are described so that the model is no strictly respected. These are discussed in detail in [17]. With regard on the format adopted for the description of security patterns, in general, it adopts the definition of a template, which contains a series of named elements and a defined scope. A good description will help to capture immediately the essence of a pattern, i.e., what is the problem to which attends and what is the proposed solution. It also offers all the details necessary to implement it and consider the consequences of its application.

It is important to note that not all fields are required. For example, in the case of some patterns is difficult or unnecessary to provide detailed descriptions of their structure, behavior and the implementation, because the information can be well integrated in the description of the solution. Likewise, in [18, 19] have defined other formats for the description of security patterns, also based on templates defined as sets of named elements and associated scope. A preliminary analysis of the different types of templates shows that there is not an exact match between the elements and scope, although all of them captured approximately the same aspects of security pattern to describe it. In [17] presented the results obtained with respect the variability of the aspects used for the formal definition of security patterns, which were obtained from a in-depth review of 364 security patterns collected from different sources, which were published during the years 1997-2012. These review also allowed us to see very high heterogeneity on the way of naming the aspects and there are no criteria of correspondence between the different names, such as exits between the descriptions of GoF and POSA [20].

2.2 Corporate and community knowledge compilation

But already there is no denying the need to address the problem of security products and services based on software, responsible for projects still face serious difficulties in addressing effectively the priorities identified in terms of security. What is the cause of this discrepancy? We cannot say that only due to ignorance about the existence of attackers who manage to exploit vulnerabilities in software. In fact, we can observe a significant disconnect between security experts and software development teams; the first are focused on the security of a system, while the seconds in building a system. For the latter, security is one of the non-functional goals with relevant, but just one of many.

Security patterns constitute a technology adopted for the description, communication and knowledge sharing, and proposed as a bridge that seeks to reduce this gap by capturing security expertise in the form of verified solutions to recurring problems. It is expected that they will be understood and used by the different members of development teams that are not security experts. Since the emphasis is on security, these patterns capture the strengths and weaknesses of different approaches in order to allow development teams to make informed decisions that balance security with other objectives.

We must emphasize that in this paper we have adopted the intelligence concept to refer to the set of practices that give rise to a collection of corporate knowledge, which is used to carry out proactive software security activities across the organization.

We decided to work in the organization and accessibility of that intelligence, seeking to generate a proposal for cataloguing security patterns. So we have considered the possibility of having resources that allow you to access a repository where reference to the set of defined security patterns in a systematic way of special importance. In this way, we hope to put at the disposal of the different actors involved in the software development process (project leaders, architects, designers, QA managers, programmers, testers) this knowledge drawn from the real world on the basis of the experience of specialists in security.

2.3 Cataloguing security patterns

At present, a multiplicity of sources exists where there are available different groups of security patterns defined throughout the time. This does that its discovery demands a level of effort that, normally, discourages to whom they are destinated to make use of them in systematic form. A centralized catalogue is a tool that acts as a starting point for the search and identification of one or more solutions to a security problem that is intended to resolve, expressed through security patterns. In this way it seeks to establish a bridge between the intelligence developed by security experts and the needs of knowledge of the software development teams.

To do this, we define as main requirement that the information offered by the catalogue be appropriate to "find" the security pattern. This information is extracted or inferred from the description of the pattern itself, and includes extensions that facilitate the categorization of the pattern according to established criteria. In the current design of the catalogue we adopted two criteria: (1) the ***security attributes*** impacted by the problem described; (2) the ***software development process phase*** to which applies the pattern. In this way, we hope that the information related with security patterns presented via the catalogue helps to capture immediately the essential aspects of them.

3 Proposal for cataloguing and its contribution

Our problem then is to define a way of structuring and indexing a catalogue of security patterns, so it could result quite easy to find a pattern that proposes a solution to an identified security situation, and from there access to the complete original reference of

the security pattern description. We define the set of attributes shown in Table 1 as the common attributes of security patterns that allow finding their references.

Table 1. Attributes for describing security patterns.

Nombre	Name of the original definition security pattern.
Objetivo	Problem that the security pattern meets. It is a response to a security problema.
Clasificación	Based on the phase of the software development process in which normally the pattern applies: requirements, analysis, design, coding, testing, implementation.
Aspecto seguridad afectado	Confidentiality, integrity, availability, accountability, non-repudiation
Keyworkds	They serve as a complementary reference to the security pattern.
Referencias	Links towards the documents and/or web pages where one finds the detailed description of the pattern.

Except for the 'Nombre' attribute, for the remaining attributes is necessary to make an analysis of the security pattern description in its original source, extract the attributes concepts associated with the selected attributes and perform the cataloguing of the security pattern. With respect to this concepts extraction process during the review of a security pattern, we found a guide in [17]; this paper summarized the results about the quality of the information included in the aspects that describe a security pattern, as well as the frequency of use of these aspects used along 67 publications of security patterns. In this regard, we can highlight the following as a guide to the extraction process:

- *Solution* (present in 87% of publications), is used to describe what security aspects attends the pattern and how it could be implemented;
- *Problem* (present in 84% of publications), in many cases includes abstract or oversimplified description of the problem;
- *Related Patterns and Consequences* (present in 75% of publications), require a good understanding of other security patterns as potential impact in the field of security so that they can be used as elements of distinction.
- *Context* (present in 49% of publications) generally includes a brief description, which makes it difficult to extract sufficient knowledge or even an idea about what the security pattern;
- *Known Use* (present in 46% of publications), described in what real-life cases you can use the pattern and provides guidance on its application domain.

4 Cataloguing process and prototype

Here by means of an example, the description of the process of incorporation of a security pattern to our catalogue. The selected pattern is called *Authenticated Session* [21], which is summarized in Figure 1. This process consists in the reading and analysis of the description security pattern conducted by a security expert, and in obtaining the data shown in Table 2, which result from the following references (the latter depends on the particular format adopted by the authors for the description of the security pattern and the quality of associated information): i) *Objetivo*: it is inferred from the Abstract section of the description; ii) *Clasificación*: the security expert who performs the reading and analysis of the security pattern inferred that the

Authenticated Session
(a.k.a. Server-Side Cookies, Single Sign-On)

Abstract

An authenticated session allows a Web user to access multiple access-restricted pages on a Web site without having to re-authenticate on every page request. Most Web application development environments provide basic session mechanisms. This pattern incorporates user authentication into the basic session model.

Problem

HTTP is a stateless, transaction-oriented protocol. Every page request is a separate atomic transaction with the Web server. But most interesting Web applications require some sort of session model, in which multiple user page requests are combined into an interactive experience. As a result, most Web application environments offer basic session semantics built atop the HTTP protocol. And the protocol itself has evolved to provide mechanisms -such as basic authentication and cookies- that allow session models to operate correctly across different Web browsers. An obvious use for session semantics is to allow users to authenticate themselves once instead of every time they access a restricted page. However, great care must be taken when using session semantics in a trusted fashion. Most session mechanisms are perfectly adequate for tracking non-critical data and implementing innocuous transactions. In such cases, if an end user circumvents the session mechanism, no harm is caused. But it is easy to make mistakes when applying session mechanisms to situations where accountability, integrity, and privacy are critical.

Solution

An authenticated session keeps track of a user's authenticated identity through the duration of a Web session. It allows a Web user to access multiple protected pages on the Web site without having to re-authenticate him/her-self on every page request. It keeps track of the last page access time and causes the session to expire after a predetermined period of inactivity. …..

Examples

Many significant Web banking and e-commerce applications rely on this pattern. Any site that enforces user authentication and does not store that information on the client uses something similar. ….

Related Patterns

· *Network Address Blacklist* – a related pattern that demonstrates a procedure for blocking a network address from further access attempts if a session identifier guessing attack is conducted.

· *Password Authentication* – a related pattern that presents the secure management of passwords, which are almost always used as the authentication mechanism for this pattern.

References

[1] Coggeshall, J. "Session Authentication". http://www.zend.com/zend/spotlight/sessionauth7may.php, May 2001.

[2] Cunningham, C. "Session Management and Authentication with PHPLIB". http://www.phpbuilder.com/columns/chad19990414.php3?page=2, April 1999.

[3] Kärkkäinen, S. "Session Management". *Unix Web Application Architectures*. http://webapparch.sourceforge.net/#23, October 2000.

Figure 1. Selected security pattern to cataloguing: *Authenticated Session* [21].

Table 2. Data for cataloguing the security pattern selected.

Nombre	**Authenticated Session**
Objetivo	Allow the access to multiple pages with restricted access, without having to re-authenticate again and again. Keep authentication information in the system's navigation
Clasificación	Design
Aspecto de seguridad afectado	Accountability, Integrity.
Keywords	Authentication, Single Sign-On
Referencias	• Security Patterns Repository v1.0.pdf • Cunningham, C. "Session Management and Authentication with PHPLIB". http://www.phpbuilder.com/columns/chad19990414.php3, (Rev. Mayo 2013). • Kärkkäinen, S. "Session Management". *Unix Web Application Architectures*. http://webapparch.sourceforge.net/#23, October 2000. (Rev. Mayo 2013)

same applies to the design phase, in the aspect of 'user authentication'; iii) *Aspecto de seguridad afectado*: in the final paragraph of section description Problem refers to "But it is easy to make mistakes when applying session mechanisms to situations where accountability, integrity, and privacy are critical", accordingly, the security pattern seeks to address these shortcomings; iv) *Keywords*: inferred from Name and Know-As-As sections; v) *Referencias*: from the references listed in the References section, selecting those sources whose availability at the time of cataloguing can be verified. Obtained these data, we proceed to cataloguing the security pattern selected for example, using the prototype that we describe in the following section.

To realize this proposal, we analyzed extended use alternative tools that allow us to its implementation. We define build a prototype using a tool for management of bibliographic references: 'JabRef'. This is a very low-cost alternative to test the idea which, if it actually works, can then be extended to massive tools, as for example a web application with features 'wiki'.

JabRef is configurable bibliographic reference management software. It use native format BibTex (a text-based and independent of the style file format) to define lists of bibliographic items, articles, thesis, etc. For the generation of the proposed catalogue we take advantage of this facility for recording references in this manager; in addition, the existence of distinct and specific attributes in each pattern allows us to generate a special entry type (Security Pattern type) with their respective fields and general and/or optional information. Another important feature of JabRef is the use of LaTeX, which allows us to transfer automatically and without major difficulties (given the use of well-defined parameters) the current catalogue to any other type of software that allows the entry of the same language: databases, Wikipedia pages, another specific or general purpose manager, etc. Figure 2 shows a screen from the JabRef interface, where:

- *Groups* (section 1). Created groups that correspond to the phases of software development in which the pattern is normally applicable. Selecting a group in section 2 (Entry) will appear the patterns that correspond to that classification.
- *Entry* (section 2). Entries loaded at the base are presented as a table with the fields as columns, which can be used to select a criterion of order by pressing the desired column. You can see all the references or related files by right clicking

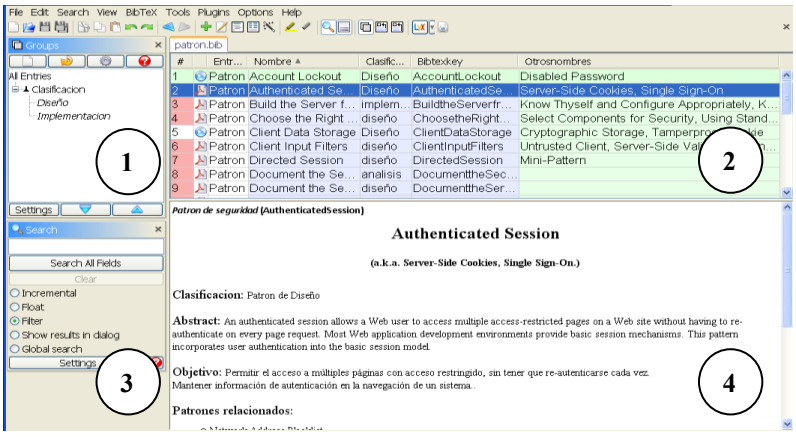


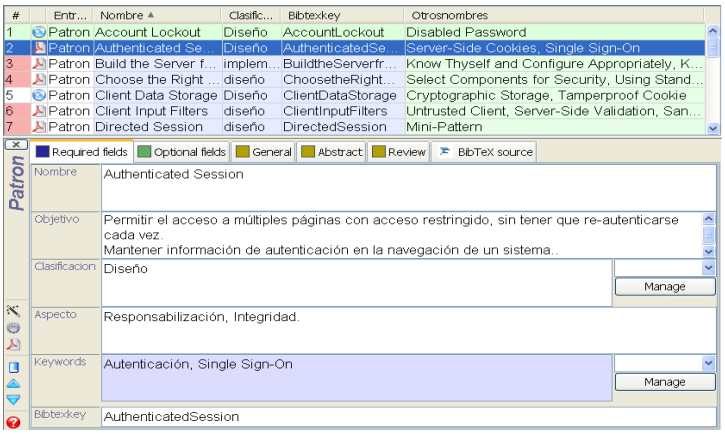
Figure 2. JabRef Catalogue Interface.

on the desired row in the column '*File*' (second column), folding a list of values. If you select one of these, will go to the required reference, whether a document available locally or on the Internet.

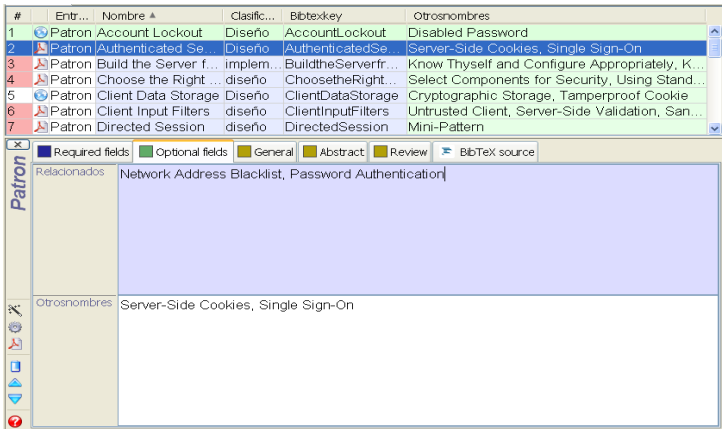
- *Search* (section 3). It allows to search according to an attribute and filters inside the entire catalogue. After a search and already showing the selected patterns that contain the phrase or word searched, when you double-click on them and navigating through the different tabs of the query, words appear highlighted in blue.
- *Preview* (Section 4). It shows the information of the security pattern selected in PDF format, which is a friendly representation of the contents of the pattern by default. By pressing the right mouse button you can print preview.

This figure shows the way how catalogued information relating to the Authenticated Session pattern [21] is displayed, where there are the attributes proposed to categorize a security pattern.

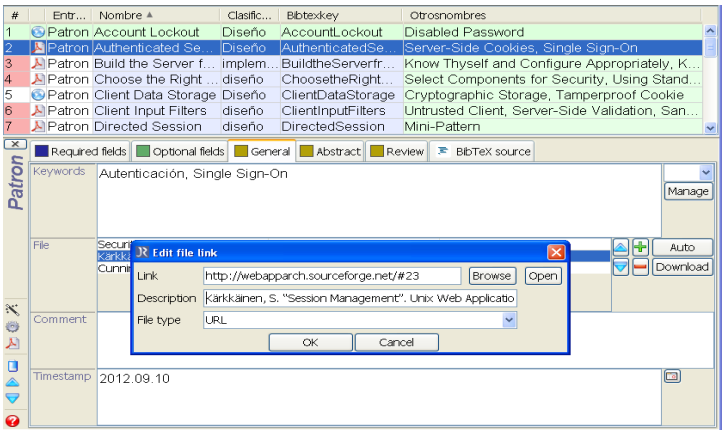
Cataloguing a security pattern: Figure 3 shows some of the facilities offered by the developed prototype to incorporate information from the attributes listed in Table 2: Figure 3.a): Loading the 'Required Fields' (mandatory): name, objective, classification, security issues, key words and Bibtexkey (a peculiarity of JabRef and



a) Required Fields.



b) Optional Fields.



c) Loading 'Keywords' and 'Referencias'.

Figure 3 Interfaces for loading data for cataloguing attributes *Authenticated Session* security pattern.

bibtex references, which is used for references to the security pattern); Figure 3b) Information is supplemented with related patterns and other well-known names (in both cases, if any); Figure 3.c): Defining 'Keywords' and 'Referencias' for the security pattern, taking advantage of the JabRef's facility for linking local and external files as 'Files'.

Searching a security pattern in the catalogue: Figure 4 shows the result of a search for references to security patterns applicable to the design phase and that attend to the confidentiality as a security attribute.

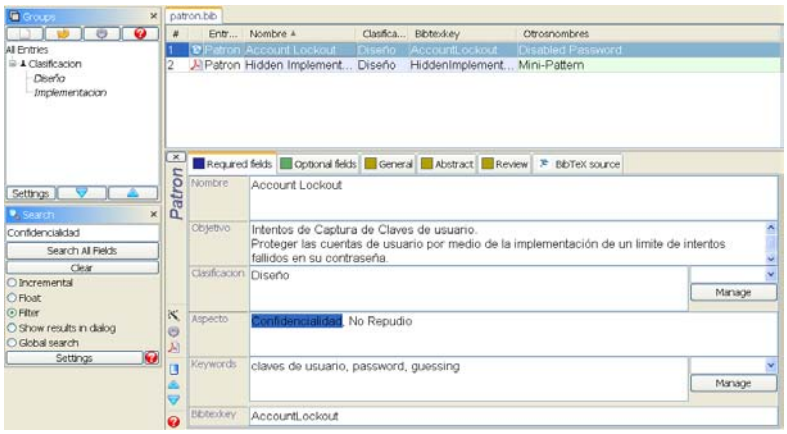


Figure. 4. Results of a search in the security pattern catalogue.

5 Conclusions and Future Work

This proposal aims to provide to the security community these quick references, and go replenishing it as it's used. The tool chosen to build a catalogue of references

helps us to test in a simple way, and eventually modify, the proposed structure and criteria, and consider the possibility of replace with another with higher performance and style. This prototype allows us to define the bases to make an extension to a wiki-style web tool, which will be available to the whole community, and also to be updated by it. In addition to the web tool, it will be necessary to propose and publish criteria to incorporate new references to our catalogue, so that all entries follow these agreed common criteria. This is the work that we intend to address in a next phase.

References

1. McGraw, G., "Software Security: Building Security In." Addison-Wedsley. EEUU. (2006).
2. Romaniz, S., "Buenas prácticas de elicitación de los requerimientos de seguridad". IV Congreso Iberoamericano de Seguridad Informática -CIBSI2007-, Argentina (2007).
3. Meier, J. et al., "Security engineering explained." (2005). Available in http://www.microsoft.com/download/en/confirmation.aspx?id=20528.
4. Castellaro, M. y otros, "Hacia la Ingeniería de Software Seguro." XV Congreso Argentino de Ciencias de la Computación, CACIC2009. Argentina. (2009).
5. Solinas, M., "Elicitación y trazabilidad de requerimientos utilizando patrones de seguridad." Universidad Nacional de La Plata. Argentina. (2012). Available in http://sedici.unlp.edu.ar/bitstream/handle/10915/421/Documento_completo.pdf?sequence=1.
6. Garvin, D., "What does product quality really mean?" Sloan Management Review, Vol 26, No 1. (1984).
7. Romaniz, S. y otros, "La seguridad como aspecto organizacional y transversal en proyectos de Sistemas de Información." 38 Jornadas Argentinas de Informática 38JAIIO. Argentina. (2009).
8. Schumacher, M. et al., "Security Patterns: Integrating Security and Systems Engineering." John Willey & Sons Inc. EEEUU. (2006).
9. Coplien, J.: "Design Pattern Definition - Software Patterns." Available in http://www.hillside.net/component/content/article/50-patterns/222-design-pattern-definition.
10. Schumacher, M.: "Security engineering with patterns-origins, theoretical model, and new applications." Springer-Verlag. (2003).
11. Alexander, C.: "The Timeless Way of Building." Oxford University Press. EEUU. (1979).
12. Yoshioka, N. et al.: "A survey on security patterns." Progress in Informatics. (2008).
13. Fernandez, E. et al. "Classifying Security Patterns." Progress in WWW Research and Development Volume 4976. (2008).
14. Washizaki, H. et al. "Improving the Classification of Security Patterns." 20th International Workshop on Database and Expert Systems Application, DEXA'09. Austria. (2009).
15. Buschmann, F. et al. "Pattern-Oriented Software Architecture: A System of Patterns." Chichester, UK: Wiley, 1996.
16. Schumacher, M. et al., "Security engineering with patterns." Proceedings of the Conference on Pattern Languages of Programs, pp. 1–17. (2001).
17. Bunke, M. et al. "Organizating Security Patterns Related to Security and Pattern Recognition Requirements." International Journal on Advances in Security, vol 5 no 1 & 2 (2012).
18. Kienzle, D.: "Security Patterns Template and Tutorial version 1.0." (2008). Available in http://www.scrypt.net/~celer/securitypatterns/template%20and%20tutorial.pdf
19. Blakley, B. et al. "Security Design Patterns." The Open Group. (2004). Available in https://www2.opengroup.org/ogsys/catalog/g031.
20. Henninger, V. et al. "Software pattern communities: Current practices and challenges." Proceedings of the Conference on Pattern Languages of Programs PLOP. New York, NY, USA. ACM, 2007, pp. 14:1–14:19.
21. Darrell M. Kienzle et al. "Security Patterns Repository Version 1.0." Available in http://www.scrypt.net/~celer/securitypatterns/repository.pdf