Issues Affecting Security Design Pattern Engineering

Jan de Muijnck-Hughes and Ishbel M. Duncan

School of Computer Science,
University of St Andrews,
St Andrews, Fife, UK
{jfdm, ishbel.duncan}@st-andrews.ac.uk

Abstract Security Design Patterns present the tried and tested design decisions made by security engineers within a well documented format. Patterns allow for complex security concepts, and mechanisms, to be expressed such that non-domain experts can make use of them. Our research is concerned with the development of pattern languages for advanced crypto-systems. From our experience developing pattern languages we have encountered several recurring issues within security design pattern engineering. These issues, if not addressed, will affect the adoption of security design patterns. This paper describes these issues and discusses how they could be addressed.

Keywords: security, design patterns, security patterns, pattern engineering, software engineering

1 Introduction

Security Design Patterns provide well described solutions to recurrent security problems [15]. By using security design patterns complex security concepts and mechanisms can be expressed concisely and explicitly such that non-domain experts can understand them, and consequently use them to address security concerns within their applications [5].

Our research investigates how a pattern-based approach can be used to address how guarantees towards encrypted access control can be made using advanced crypto-systems by non-experts. The main result of our investigation is the creation of a pattern language detailing both the crypto-system itself and its deployment. During the development of our pattern language several areas of concern became apparent that adversely affected our experience. These issues pertain to: (a) Pattern Development (Section 4); (b) Pattern Templates (Section 5); (c) Pattern Encoding (Section 6); (d) Pattern Classification (Section 7); (e) Pattern Repositories (Section 8); and (f) Pattern Evaluation (Section 9). It is our belief that if these issues are left unresolved they may have a negative effect upon both the development of security design patterns but also the adoption of pattern-based approaches when developing secure applications.

Proceedings of Cyberpatterns 2013, pp 54–61

This paper presents an overview of the perceived issues. We begin first with some preliminary information over Security Patterns (Section 2), and Security Pattern Engineering—Section 3. The remainder of the paper details the perceived issues.

2 Security Design Patterns

Within the domain of software engineering patterns represent a well documented solution to a recurrent problem within a particular context. Typically, patterns represent frequently recurring structures, behaviours, activities, and processes used to solve recurrent problems, and are used to document the *tried and tested* design decisions made by software developers and engineers when providing a solution [8].

Consequently a pattern will encapsulate the engineer's experience within the pattern itself; patterns will embody expert domain knowledge. Patterns provide a *separation of concern* between solution conception and solution application. By using patterns not only is expert domain knowledge made more accessible to non-domain experts but cross domain knowledge transfer can be established.

Pattern templates are a set of predefined headings used to document individual patterns. These headings are used to capture the various different aspects of the problem and solution being presented. Typically, the aspects described within a pattern will include: (a) the context in which the pattern is being applied; (b) the problem the pattern is solving; (c) forces that affect not only how the problem can be solved but also how the solution is implemented; (d) the solution presented by the pattern; (e) the solution's structure and dynamics; and (f) guidance for implementing the pattern. Patterns are themselves written using a mixture of natural language descriptions and formal models to document the solution's components and their interaction.

First introduced in Yoder and Barcalow [20], *Security Design Patterns* are patterns describing solutions to recurring security related problems. Not only can security design patterns be used to describe well known security concepts but also be used to describe various security mechanisms used within software engineering [15]. Pattern-oriented approaches that use security design patterns are increasingly being used to develop secure systems [7]. Security pattern languages are pattern languages within the security domain. By using security patterns, complex security concepts and mechanisms can be expressed concisely and explicitly such that non-domain experts can understand them, and consequently use them [5]. When designing security systems abstract patterns [6] can be used to abstract over multiple similar patterns that address a common security problem. According to Bunke et al. [1] there are 409 known security design patterns, providing solutions to security problems as diverse as access control, session management, and identity management. These patterns were collected through a systematic literature review of patterns published between the 1997 and 2010. The relevant literature documenting these patterns can be found within Bunke et al. [1].

Remark 1. Security Design Patterns describe how security problems can be addressed. The antithesis of security design patterns are known as *Attack Patterns*. Attack patterns describe how security problems can be caused [16]. In this paper we are concerned with issues affecting security design patterns, only.

3 Security Pattern Engineering

The lifecycle of a pattern can be viewed as two distinct processes. The first: **Pattern Development** is concerned with the development of the pattern itself; and the second **Pattern Application** details the application of said pattern. Pattern engineering involves the steps required within these processes that allow for pattern creation and pattern deployment. Within Yoshioka et al. [21] the author's enumerate this pattern engineering process as follows:

- **Development**
  - D1: Finding a pair of recurring problem and its corresponding solution from knowledge and/or experiences of software development.
  - D2: Writing the found pair with forces in a specific pattern format.
  - D3: Reviewing and revising the written pattern.
  - D4: Publishing the revised pattern via some public or private resource (WWW, book or paper,...).
- **Application**
  - A1: Recognizing context and security problems in software development.
  - A2: Selecting software patterns that are thought to be useful for solving the recognized problems.
  - A3: Applying the selected patterns to the target problem.
  - A4: Evaluating the application result.

4 Pattern Development

Our experience stems from the creation of pattern languages and thus the development of (Security) Design Patterns. Specifically, our research requires the management and implementation of Steps D2–4 from Section 3. Although, there is existing guidance concerning how to approach pattern writing (see Wellhausen and Fießer [19] and Meszaros and Doble [14]), the authors do not provide guidance as to how this process should be managed and also implemented. Furthermore, the existing guidance has been written to aid in the creation of generic patterns, and is not specific to the creation of security design patterns.

For new pattern writers this raises the question over how *best* this development process should be managed, followed, and implemented. Several of the problems affecting pattern development are practical in nature, and relate to practical matters such as tool selection. For example, how patterns are encoded and stored during development will have an effect on how the development process can be managed and *vice versa*. Similarly, how patterns are stored once development has been completed will affect how patterns are subsequently published.

5 Pattern Templates

For usability, patterns are often described using a mixture of structured headings, textual descriptions, and formal models. Pattern templates represent a fixed set of headings that are used to describe like patterns. However, there appears to be no one common pattern template that is used through out the security design pattern community [1, 21]. Moreover, various templates have been developed for different security problem domains but are not strictly adhered to—see Bunke et al. [1]. Put succinctly the universe of security design pattern templates is too heterogeneous. Examples of different templates can be found in Buschmann et al. [2], Gamma et al. [8], Schumacher et al. [15] and Cuevas et al. [4].

Moreover, different headings within pattern templates can be used to describe similar aspects. This results in templates that have different headings but nonetheless describe the same content. Furthermore, headings within a template may be required or optional, not all headings are used. This makes both the identification of like patterns, and pattern evaluation, classification and comparison a much more difficult task to perform. When developing patterns a developer should seek to select a suitable pattern template that all the developed patterns must adhere to.

6 Pattern Encoding

Patterns are expected to be published within a variety of end media. For example, patterns can be expected to be found within academic articles, textbooks for education, and online repositories. The concern here is related to how *best* security patterns be represented (encoded) such that they can be stored and used, during pattern development and pattern application.

Outside the security design pattern area, existing efforts have utilised custom mark-up languages. For example: Entity Meta-specification Language (EML) is an XML based construct that was designed with the purpose of: *describing all kinds of software patterns and supporting concepts* Welicki et al. [18]. In a related effort Pattern Language Markup Language (PLML) and eXtended Pattern Language Markup Language (xPML) are other XML derived languages developed for the domain of HCI [12]. Lastly, the Common Attack Pattern Enumeration and Classification (CAPEC) repository [3] also utilises a custom XML schema to represent attack patterns. Should like schema be developed for security design patterns?

In general, the issues that need to be taken into account when looking to encode patterns include: the pattern template being used; the use of natural language to describe the pattern; pattern metadata; and the modelling language used to represent the solution. An encoding needs to be chosen/developed that facilitates pattern: (a) creation; (b) storage; (c) comparison (d) exportation; and (e) publication.

A related issue to pattern encoding is the representation of the prescribed solution. With design patterns, the solution's structure and dynamics are often

modelled using the Unified Modelling Language (UML) modelling language. Structure using class diagrams, and dynamics using sequence diagrams. However, UML is a graphical-oriented modelling language, the result is a series of diagrams. This is not ideal when working with the resulting models, the modelling information is lost as a result of the graphical notation. How best should the solution be modelled and presented within the pattern itself?

7 Pattern Classification

As more patterns are being developed it is important to consider how security patterns (and pattern languages) can be classified. Classification allows for patterns to be grouped according to characteristics and properties that they share. This is beneficial when developers are exploring the pattern landscape for existing patterns during pattern application [7]. Classification helps reduce the search space. Much work has already been performed in classifying security patterns, most recently by Bunke et al. [1]. The problem of pattern classification can be broken down into: (a) the development of a pattern taxonomy; and (b) the classification of a group of patterns. However, the process of developing (automatic) classification techniques, taxonomies, and performing pattern classification is made harder due to the use of non-standard pattern templates, and lack of central pattern repository. Other problems related to pattern classification include the selection of classification criteria and the visual representation of multi-dimensional criteria—see Bunke et al. [1].

8 Pattern Repositories

Pattern Repositories are centralised locations that developers can use to access patterns. Example pattern repositories can be found: online [17]; within books e.g. Schumacher et al. [15]; or (and most commonly) within academic literature. A comprehensive list of academic literature featuring security patterns can be found in Bunke et al. [1]. Pattern repositories have also been presented as a single PDF document Kienzle et al. [11]. However, these resources are either: a) incomplete; b) cannot be modified; or c) cannot be used programmatically. The CAPEC repository [3] is a good example of a pattern repository, however, it has been designed for attack and not security design patterns.

Central to security pattern engineering, and also research, is the creation of an easily accessible security design pattern repository that can be used by researchers and developers alike. The existence of such a pattern repository would provide pattern researchers with a catalogue through which they can perform pattern related research. This would also benefit pattern developers. For software developers, a centralised repository will facilitate access to a variety of security design patterns that they can examine/select for their needs during pattern application. When looking to develop such a repository for security design patterns the CAPEC repository appears to be a good place to start.

9 Pattern Evaluation

Pattern evaluation is one of the lesser reported aspects within security pattern research. A well known practise within the pattern community is that of *Peer Review* and *Shepherding* [19]. During the pattern engineering process shepherding pairs the pattern author with another, experienced, pattern writer who provides authoritative guidance and advice concerning the pattern development. However, an inherent problem with shepherding is that the process provides subjective evaluation over the quality of the pattern itself and not the solution being described. A more formal approach to evaluation is required.

Within the security design pattern community there have been several papers that look towards security pattern evaluation [9, 10, 21]. However, each of these papers provides not only a different evaluation criteria but not all were designed to evaluate security patterns. Analyses were given of the pattern landscape (at the time) as well. This raises the question concerning which properties of a security pattern should be selected for evaluation. Heyman et al. [10] examined a pattern according to the appropriateness and quality of documentation. Yoshioka et al. [21] examined patterns according to a patterns ease of use; effectiveness; and sufficiency. Are these existing properties sufficient, or should other properties, for instance usability, be examined?

When looking to establish an evaluation framework for security design patterns an evaluation approach needs to be defined. Should the approach be quantitative, qualitative, or a mixture of both? Regardless of the approach taken, the selected criteria should assess the patterns themselves and not the actual implementation [9].

The precise nature concerning how security patterns should be evaluated is still ongoing. However, work by Laverdière et al. [13] that utilises the *House of Quality* evaluation framework does look most promising.

10 Conclusion

Although the area of security patterns is not new, there are several worrying trends seen within security pattern research. For further research into, and development of, security design patterns these areas of concern need to be addressed.

There are a variety of pattern templates used by pattern developers. Some templates are unique to the pattern itself, while others are variations of existing ones. With the rich variety of templates used this makes the classification, identification and comparison of like security patterns more difficult. Standard templates need to be defined for key domain areas, and adopted by all pattern developers within those domains.

Pattern development has also been made more difficult due to the lack of a central pattern repository. This lowers the accessibility of the patterns themselves. Not all developers will have access to academic literature. Pattern developers, and users, would benefit greatly from a centralised repository.

Perhaps the most striking element is the lack of formal evaluation in relation to the patterns themselves, and the solutions represented therein. Patterns are

supposed to represent the *tried and tested* design decisions made by software developers/engineers when providing a solution. How can a developer know that the solution presented is 'good'? Can trust be established in relation to the quality of the pattern? To promote the adoption of patterns a more formal evaluation framework needs to be established and made accessible. How this framework should look is open research.

Bibliography

[1] Bunke, M., Koschke, R., Sohr, K.: Application-domain classification for security patterns. In: PATTERNS 2011, The Third International Conferences on Pervasive Patterns and Applications. pp. 138–143. ThinkMind (2011)

[2] Buschmann, F., Henney, K., Schmidt, D.: Pattern-oriented software architecture: On patterns and pattern languages. Wiley series in software design patterns, John Wiley & Sons (2007)

[3] Corporation, M.: Common Attack Pattern Enumeration and Classification Repository. Online (2013), http://capec.mitre.org/

[4] Cuevas, A., Laube, A., Sorniotti, A., Khoury, P.E., Gomez, L.: Security patterns for untraceable secret handshakes with optional revocation. International Journal On Advances in Security 3(1&2), 68–79 (sept 2010)

[5] Delessy, N., Fernandez, E., Larrondo-Petrie, M.: A pattern language for identity management. In: Computing in the Global Information Technology, 2007. ICCGI 2007. International Multi-Conference on. p. 31 (march 2007)

[6] Fernandez, E.B., Washizaki, H., Yoshioka, N.: Abstract security patterns. In: Proceedings of the 15th Conference on Pattern Languages of Programs. pp. 4:1–4:2. PLoP '08, ACM, New York, NY, USA (2008), http://doi.acm.org/10.1145/1753196.1753201

[7] Fernández, E.B., Yoshioka, N., Jürjens, H.W.J., Hilst, M.V., Pernul, G.: Using Security Patterns to Develop Secure Systems, chap. 2, pp. 16–31. IGI Global (2011)

[8] Gamma, E., Helm, R., Johnson, R., Vlissides, J.: Design Patterns: Elements of Reusable Object-Oriented Software. Addison-Wesley (1994)

[9] Halkidis, S., Chatzigeorgiou, A., Stephanides, G.: A qualitative evaluation of security patterns. In: Lopez, J., Qing, S., Okamoto, E. (eds.) Information and Communications Security, Lecture Notes in Computer Science, vol. 3269, pp. 251–259. Springer Berlin / Heidelberg (2004), http://dx.doi.org/10.1007/978-3-540-30191-2_11

[10] Heyman, T., Yskout, K., Scandariato, R., Joosen, W.: An analysis of the security patterns landscape. In: Proceedings of the Third International Workshop on Software Engineering for Secure Systems. p. 3. SESS '07, IEEE Computer Society, Washington, DC, USA (2007), http://dx.doi.org/10.1109/SESS.2007.4

[11] Kienzle, D.M., Elder, M.C., Tyree, D., Edwards-Hewitt, J.: Security patterns repository version 1.0 (2003), http://scrypt.net/~celer/securitypatterns/repository.pdf, online [Accessed 2012-02-27]

[12] Kruschitz, C., Hitz, M.: Bringing formalism and unification to human-computer interaction design patterns. In: Proceedings of the 1st International Workshop on Pattern-Driven Engineering of Interactive Computing Systems. pp. 20–23. PEICS '10, ACM, New York, NY, USA (2010), http://doi.acm.org/10.1145/1824749.1824754

[13] Laverdière, M.A., Mourad, A., Hanna, A., Debbabi, M.: Security design patterns: Survey and evaluation. In: Electrical and Computer Engineering, 2006. CCECE '06. Canadian Conference on. pp. 1605–1608 (2006)

[14] Meszaros, G., Doble, J.: Pattern languages of program design 3. chap. A pattern language for pattern writing, pp. 529–574. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA (1997), http://dl.acm.org/citation.cfm?id=273448.273487

[15] Schumacher, M., Fernandez-Buglioni, E., Hybertson, D., Buschmann, F., Sommerlad, P.: Security patterns: integrating security and systems engineering. Wiley series in software design patterns, John Wiley & Sons (2006)

[16] Sethi, A., Barnum, S.: Introduction to attack patterns. Tech. rep., Cigital Inc. (Nov 2006), https://buildsecurityin.us-cert.gov/articles/knowledge/attack-patterns/introduction-to-attack-patterns

[17] The Hillside Group: Patterns catalog. Online (2013), http://hillside.net/patterns/patterns-catalog

[18] Welicki, L., Manuel, J., Lovelle, C., Aguilar, L.J.: Patterns meta-specification and cataloging: towards knowledge management in software engineering. In: Companion to the 21st ACM SIGPLAN symposium on Object-oriented programming systems, languages, and applications. pp. 679–680. OOPSLA '06, ACM, New York, NY, USA (2006), http://doi.acm.org/10.1145/1176617.1176670

[19] Wellhausen, T., Fießer, A.: How to write a pattern? a rough guide for first-time pattern authors. In: Proceedings of the 16th European Conference on Pattern Languages of Programs. pp. 5:1–5:9. EuroPLoP '11, ACM, New York, NY, USA (2012), http://doi.acm.org/10.1145/2396716.2396721

[20] Yoder, J., Barcalow, J.: Architectural patterns for enabling application security. In: Proceedings of the Conference on Pattern Languages of Programs (PLoP 1997). Monticello/IL (1997)

[21] Yoshioka, N., Washizaki, H., Maruyama, K.: A survey on security patterns. Progress in Informatics (5), 33–47 (2008)