Abstract security patterns

Eduardo B. Fernandez
Florida Atlantic University
US
ed@cse.fau.edu

Hironori Washizaki
Waseda University
Japan
washizaki@waseda.jp

Nobukazu Yoshioka
Natonal Institute of Informatics
Japan
nobukazu@nii.ac.jp

ABSTRACT

We introduce the concept of "abstract" security patterns that deal with abstract security mechanisms, rather than concrete implementations. We also show an organization of abstract security patterns and concrete ones into hierarchies.

Categories and Subject Descriptors

D.2.11 [**Software Engineering**]: Software Architectures—*Patterns*

General Terms

Design, Documentation, Experimentation, Security

Keywords

Patterns, Classification, Modeling

1. INTRODUCTION

We can think of building a software system as solving a problem. In the analysis stage of the development, we are trying to make the problem precise; we are not concerned with software aspects such as implementation and platform. From a security point of view, we only want to indicate which specific security mechanisms are needed, not their implementation. Therefore we need at this stage a set of patterns that defines abstract security mechanisms. These patterns should specify only the fundamental characteristics of the mechanism or service, not specific software aspects. Most works on security patterns[7, 8] emphasize concrete patterns that solve security problems at given architectural levels or units, e.g., secure VAS in operating systems[3]. In fact, we have not seen any work where this abstraction level is explicitly considered.

We present here the concept of abstract security patterns based on the considerations above and we show some examples. The common context of all abstract security patterns is the problem space. We relate them to each other using pattern diagrams based on the problem space. We also relate them to architectural (software oriented) security patterns.

Some of these patterns correspond to basic security mechanisms, e.g., Access control (such as `Authorization`), `Security Logger`, and `Authenticator`. Others specify more detailed aspects, e.g., Access Control/Authorization models include the `Access Matrix`, `Role-Based Access Control (RBAC)`, `Multilevel Security`, and `Attribute-Based Access Control (ABAC)` models.

They should not be confused with patterns that describe basic security principles, e.g., `Single-Point-of-Access`[10]. Abstract security patterns correspond to mechanisms or services, not principles.

2. ABSTRACT PATTERNS

Figure 1 is a pattern diagram[7] consisting of eight security patterns and relations among them to show how abstract patterns are related to other patterns. Pattern `Credential` represents some aspect of a conceptual model, and the basic security services are described by patterns `Authenticator`[2], `Authorization`[1, 7] and `Security Logger`[8].

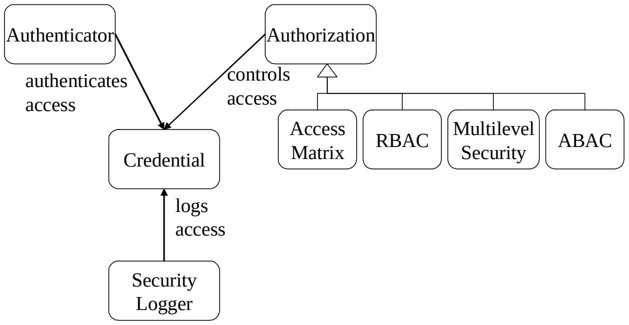

Figure 1: Basic security services

While the abstract security patterns exist for these services, we need to revisit them to emphasize their fundamental properties. `Credential` provides secure means of recording authentication and authorization information for use in domains where we are not known. Moreover in Figure 1, the following security patterns deal with more concrete solutions for the same purpose of `Authorization` (i.e., controlling accesses):

- `Access Matrix`. Assign rights to individual subjects, e.g., users.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission. A preliminary version of this paper was presented in a writers' workshop at the 15th Conference on Pattern Languages of Programs (PLoP).
PLoP '08, October 18–20, 2008, Nashville, TN, USA.
Copyright 2008 is held by the author(s). ACM 978-1-60558-151-4.

- Role-Based Access Control (RBAC)[7]. How do we assign rights to people based on their functions or tasks? Assign people to roles and give rights to these roles so they can perform their tasks.
- Multilevel Security[1, 7]
- Attribute-Based Access Control (ABAC)[6]. Allow access to resources based on the attributes of the subjects and the properties of the objects.

In the figure, we represented these specialization relations as generalization-specialization relationships in the form of UML-like class diagram. We will introduce such relations in detail in the next section.

3. USE OF ABSTRACT PATTERNS

Possible uses for these patterns include:

- Guide the search for new patterns. An abstract pattern defines a range of patterns and one can see if corresponding patterns exist at the lower levels. Moreover such relations might form a pattern language.
- Serve as abstract prototypes for similar concrete patterns. Starting from an abstract pattern it is easy to see what happens at a specific architectural level.
- Serve as ways to connect and relate different sets of patterns. For example, a Communication Channel can use Intrusion Detection.

We can make generalization hierarchies with patterns[9] and define patterns that are more and more concrete. For example, starting from a Communication Channel pattern, a Secure Channel denotes a channel where some security measure has been applied.

These patterns are the roots of pattern hierarchies where each lower level is a pattern specialized for some specific context. That is, the context is one of the main determinants of the difference of a pattern with another. The context defines the environment where the pattern applies. In general, the context of a lower-level pattern includes the context of its ancestors: $C_i \subset C_j$, where $i < j$ in the hierarchy, where C_x denotes the context of a pattern x . For example, the context of an abstract Credential applies to any distributed domain while the context of an X.509 certificate applies only to distributed systems that follow this standard.

The reverse is true about forces and consequences, the forces in a concrete pattern include what is in the abstract pattern plus new forces (and their consequences) due to the more specific environment. Their threats are specific versions of the abstract pattern threats.

We can draw pattern hierarchies showing several levels in one diagram as in Figure 2. Alternatively, we can draw separate graphs for each level. The first type is useful when we want to correlate patterns at different abstraction levels or we want to understand or explain a complete system. The second type is better when we are working at a specific level, e.g., designing an operating system[3, 4].

4. CONCLUSION

We have introduced the concept of abstract security pattern and shown that it has several advantages, including providing insight into the nature of security patterns. Future work includes generalizing these ideas to other types of patterns.

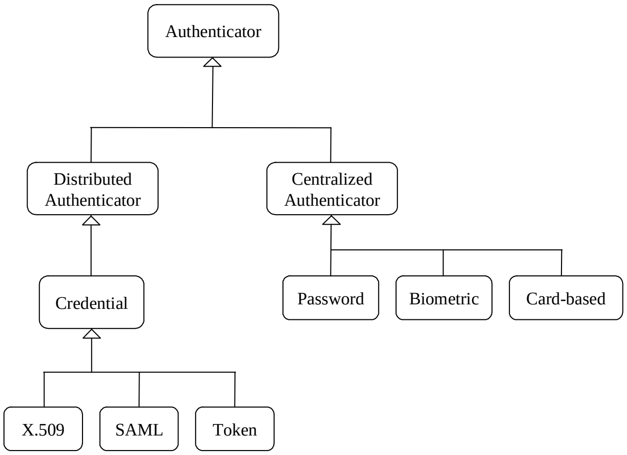

Figure 2: The authentication hierarchy

5. REFERENCES

[1] E B. Fernandez and R.Y. Pan, "A pattern language for security models", Procs. of PLoP 2001, http://jerry.cs.uiuc.edu/ plop/plop2001/ accepted_submissions/accepted-papers.html

[2] E.B.Fernandez, "Patterns for operating systems access control", Procs. of PLoP 2002, http://jerry.cs.uiuc.edu/ plop/plop2002/proceedings.html

[3] E.B.Fernandez and J.C.Sinibaldi, "More patterns for operating systems access control", Procs. EuroPLoP 2003.

[4] E. B. Fernandez and R. Warrier, "Remote Authenticator/Authorizer", Procs. of PLoP 2003.

[5] P. Morrison and E.B.Fernandez, "The Credential pattern", Procs. of PLoP 2006.

[6] T. Priebe, E.B.Fernandez, J.I.Mehlau, and G. Pernul, "A pattern system for access control", Procs of the 18th. Annual IFIP WG 11.3 Working Conference on Data and Applications Security, 2004.

[7] M. Schumacher, E. B.Fernandez, D. Hybertson, F. Buschmann, and P. Sommerlad, Security Patterns: Integrating security and systems engineering", Wiley 2006.

[8] C. Steel, R. Nagappan, and R. Lai, Core Security Patterns: Best Strategies for J2EE, Web Services, and Identity Management, Prentice Hall, Upper Saddle River, New Jersey, 2005.

[9] H. Washizaki, E. B. Fernandez, K. Maruyama, A. Kubo, and N. Yoshioka, "Improving the Classification of Security Patterns", Procs. of 3rd International Workshop on Secure systems methodologies using patterns (SPattern '09), 2009. (to appear)

[10] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security". Procs. PLOP '97, Also Chapter 15 in Pattern Languages of Program Design, vol. 4 (N. Harrison, B. Foote, and H. Rohnert, Eds.), Addison-Wesley, 2000.