Developing a Security Patterns Repository for Secure Applications Design

Lazaros Gymnopoulos[1], Maria Karyda[1], Theodoros Balopoulos[1], Stelios Dritsas[2], Spyros Kokolakis[1], Costas Lambrinoudakis[1] and Stefanos Gritzalis[1]*

[1]Laboratory of Information and Communication Systems Security (Info-Sec-Lab), Department of Information and Communication Systems Engineering, University of the Aegean, Samos, GR-83200, Greece

{lazaros.gymnopoulos, mka, tbalopoulos, sak, clam, sgritz}@aegean.gr

[2]**Department of Informatics, Athens University of Economics and Business, GR-10434, Greece**

sdritsas@aueb.gr

*Corresponding author: Stefanos Gritzalis, Associate Professor,
Head, Dept. of Information and Communication Systems Engineering,
University of the Aegean, Karlovassi, Samos, GR-83200, GREECE,
Tel: +30-22730-82234, Fax: +30-22730-82009, e-mail: sgritz@aegean.grl

Abstract: Application developers are often confronted with difficulties in choosing or embedding security mechanisms that are necessary for building secure applications, since this demands possessing expertise in security issues. This problem can be circumvented by involving security experts early in the development process. This practice, however, entails high costs; moreover communication between developers and security experts is usually problematic and security expertise is difficult to be captured and exploited by developers. This paper proposes that the process of building secure applications can be facilitated through the use of security patterns. It presents a security patterns repository that can provide developers with an effective mechanism to address the issue of incorporating security requirements and mechanisms in application development. The paper also specifies a list of patterns and describes their basic elements. For describing and managing the patterns, the paper proposes a structure that is especially suitable for the case of security patterns. The method followed for developing the security patterns repository entails the employment of a security ontology. Finally, the paper presents a set of exemplary cases where the repository can support the software development process. The paper's contribution is an enhanced security patterns repository that allows application developers to benefit from the accumulated knowledge and expertise in the area of security, so that they are able to develop secure applications.

Keywords: Security patterns repository, application development, security ontology

1. Introduction

Building secure applications is a complex and demanding task developers often face. Meeting the specified security requirements, or embedding security mechanisms, however, is a process that entails expertise in the area of security, which most of the time software developers do not posses. Therefore, security experts often have to be involved during application development. This strategy entails high costs for software development; moreover the communication between developers and security experts is seldom unobstructed.

This paper suggests a different strategy for incorporating security in application development. It advocates the use of security patterns, by proposing a security patterns repository. The paper also addresses the issue of the limited usability of security patterns in software development, by customizing the patterns' structure so as to include security specific properties, such as threats and vulnerabilities. Thus, this paper aims to

- propose an enhanced structure for security patterns,
- describe a repository for security patterns, and

- illustrate how this repository can be deployed to support secure application development.

The next section elaborates on security patterns and their use. Section three reports on the method used for designing the structure of the security patterns that comprise the repository. The patterns repository is described in section four. Finally, the last section presents the conclusions and directions for future research.

2. Security patterns in software development

Gamma (1993) introduced patterns and since then their application in software development has continuously grown (Rosengard and Ursu 2004; Gamma, 2001). This section elaborates on *software patterns*, and particularly on *security patterns*. The typical structure of a security pattern is presented, as well as the most significant problems developers encounter in applying them.

2.1 Software Patterns

Software patterns are a solution to recurring software development problems in a specific context. They use existing, well-proven experience in software development and help promote effective software design practices. Each pattern deals with a specific, recurring problem in software design and can be used to build applications with specific properties. When software designers work on a particular problem, they often recall a related problem they have already solved, and reuse the essence of its solution to counter the new problem. Thus, "a pattern for software architecture describes a particular recurring design problem that arises in specific design contexts, and presents a well-proven generic scheme for its solution" (Buschmann et al. 1996). The solution scheme is specified by describing its constituent components, the existing relationships, and the ways in which they collaborate.

Patterns can provide many advantages for the systematic and effective development of high-quality applications with defined functional and non-functional requirements. In general, the most important advantages of using patterns in software engineering, as listed in (Buschmann et al. 1996), include the following:

- They complement existing problem-independent software development processes and methods with guidelines for solving specific recurring design and implementation problems.
- They help with the recognition of common techniques and approaches, so that high-level relationships between software systems can be used and new applications can be built as variations on old systems.
- They provide support for finding an appropriate architecture for designing software applications.
- They provide support for making right choices among design alternatives.
- They support the analysis and description of high-level properties of complex software systems.
- They can support the process of maintaining existing software systems.

2.2 Designing a Pattern System

A *pattern system* is defined as *a collection of patterns for software architecture, together with guidelines for their implementation, combination and practical use in software development*. For security patterns to be effectively used, a concise categorization within each pattern system must exist. To build the security patterns repository we followed the Pattern-Oriented Software Architecture (POSA) categorization for software patterns, proposed by Schumacher (2003). Based on this categorization, the repository consists of three different types of patterns:

- *Architectural patterns, which* refer to the high level software development process. These patterns are associated with the fundamental structural organization schema for software systems.
- *Design patterns* that refine the components of an application as well as the relationships between them.

- *Idioms, which* are patterns at the lowest level and are related and affected by the programming language that is used each time.

2.3 Security Patterns

In parallel with the evolution of software patterns, the concept of *security patterns* was introduced in order to incorporate security techniques and best practices (e.g. specific countermeasures) into the software development process. Yoder and Barkalow, as cited in (Schumacher 2003), were the first to write about security patterns. Their focus was on patterns that would enable security at the level of general software applications. Security patterns increase security know-how and awareness and can help software designers incorporate security mechanisms into software development, by using known solutions and practices in recurring security problems.

A security pattern can be defined as a particular recurring security problem that arises in a specific security context, and presents a well-proven generic scheme for its solution (Schumacher 2003). Application of security patterns can help bridge the gap between security professionals and system developers. While the emphasis is on security, these patterns capture the strengths and weaknesses of different approaches in order to allow developers to make informed trade-off decisions between security and other requirements. Therefore, security patterns can assist developers implement effective security solutions (e.g. specific security countermeasures) and use them "in a right way". Security patterns are described by using a set of predefined elements, which compose the structure of the pattern, and their values. Currently, various templates defining structures for patterns exist; they vary depending on whether they refer to design patterns, security patterns or other types of patterns (Buschmann et al. 1996, Schumacher 2003). Table 1 lists the fundamental structure most commonly used for describing and developing security patterns, as proposed in (Schumacher 2003, Kienzle et al. 2005).

Table 1: A typical structure for security patterns (based on Schumacher 2003, Kienzle et al. 2005)

Element	Description
Name	It denotes the name of the specific pattern, which should be easy to remember and indicative.
Security Context (a.k.a. Motivation)	The security context describes the conditions under which the security problem occurs. It also indicates the applicability of the security pattern. Often, the security context is introduced with the help of a scenario.
Security Problem	This element defines the security problem that occurs in the specified security context, which will be confronted by the security pattern. In the area of security a problem occurs whenever a system is not well protected.
Security Solution	This element refers to specific countermeasures, mechanisms, techniques and approaches that can be used in order to address or mitigate the security problem.
Forces	This element defines the types of trade-offs that must be considered in the presence of conflicts they might create. It should be described how they interact and differ with one another, and the goals that should be achieved.
Related Patterns (a.k.a. Security Pattern Relations)	Usually, there are security patterns that can be used as solutions to the same (or similar) problem in a different security context. Such relationships provide linkage to subsequent patterns of a collection of patterns.

It should be noted that there are also other elements that can be used for describing different properties of a security pattern. These elements, such as *referred examples*, *resulting context*, *rationale*, or *known uses*, can also be included in the structure of a security pattern. Table 2 presents an exemplary security pattern, the "*encrypted storage pattern"*, which is structured according to the properties presented in Table 1.

Table 2: An example of a security pattern example: the Encrypted Storage pattern

Element	Description
Name	*Encrypted Storage*
Security Context	Web applications are often required to store sensitive user information, such as credit card numbers, passwords etc. Although efforts can be made to protect the Web server, one can never eliminate the possibility of a new vulnerability appearing, leading to the compromise of the server.
Security Problem	The lack of specific encryption mechanisms in order to protect personal/sensitive user data and/or application components.
Security Solution	The Encrypted Storage pattern encrypts sensitive user data before it is ever committed to disk. Before it can be used, this data is decrypted in memory. If the Web server is compromised, an attacker may be able to steal the data store, but will not be able to gain access to the sensitive data.
Forces	Never attempt to invent an encryption algorithm. Use a tested algorithm from applied cryptography. If possible, use a freely available library rather than coding one from scratch. After sensitive data is used, the memory variables containing it should be overwritten. Care must be taken to ensure that sensitive data is not written into virtual memory during processing. Protection of the key. Variation: One key per user. Possible attacks.
Related Patterns	Pattern X: A related pattern that verifies all data coming from the client.

3. Designing a security patterns repository

Although security patterns can provide an answer to the problem of incorporating security requirements and mechanisms into applications, up to now they have been used mostly in an ad hoc way, resulting in inconsistencies between business objectives, security requirements and the patterns themselves. Moreover, security patterns have been used in ways that failed to establish effective communication between security experts and the software developers.

In order to address the need for incorporating security patterns into software development in a productive and consistent way, this paper proposes the use of a security patterns repository. This section describes the method followed for designing and developing the security patterns repository, while the next section presents some of the patterns comprising the repository.

3.1 Method of work

Prior to designing the repository of security patterns, we first had to decide on their structure. In order to develop a security patterns' structure that would accommodate the security related requirements of using patterns, we developed a security ontology, based on the one presented in (Dritsas et al. 2005). An ontology is *a logical theory accounting for the intended meaning of a formal vocabulary. The intended models of a logical language using such a vocabulary are constrained by its logical commitment. An ontology indirect reflects this commitment (and the underlying conceptualization) by approximating these intended models* (Mekhilef 2003). Thus, an ontology is the attempt to express an exhaustive conceptual scheme within a given domain, typically a hierarchical data structure containing all the relevant entities, their relations and the rules within that domain.

The ontology developed, depicted in Figure 1, aimed to (a) capture and express the most important security concepts for application development, (b) describe the relations among these concepts, (c) provide a common understanding and vocabulary of security issues among application developers, and (d) facilitate the development of secure applications.

The most important concepts for secure application development that were identified, as shown in Fig. 1 include the concepts *stakeholder, objective, threat, countermeasure, vulnerability, attacker etc.* These concepts are further explained in (Dritsas et al. 2005). According to the ontology that was developed, “Stakeholders” set the application “Objectives”, which are affected by “Threats”. “Assets”, owned by “Stakeholders”, have “Vulnerabilities” and are threatened by “Threats”, but they can be protected through the use of “Countermeasures” that are implemented by “Stakeholders”. A “Vulnerability” is a weakness of an “Asset”, whereas some “Threats” can be regarded as “Deliberate Attacks” that are launched by “Attackers”, through exploiting the assets’ “Vulnerabilities”.

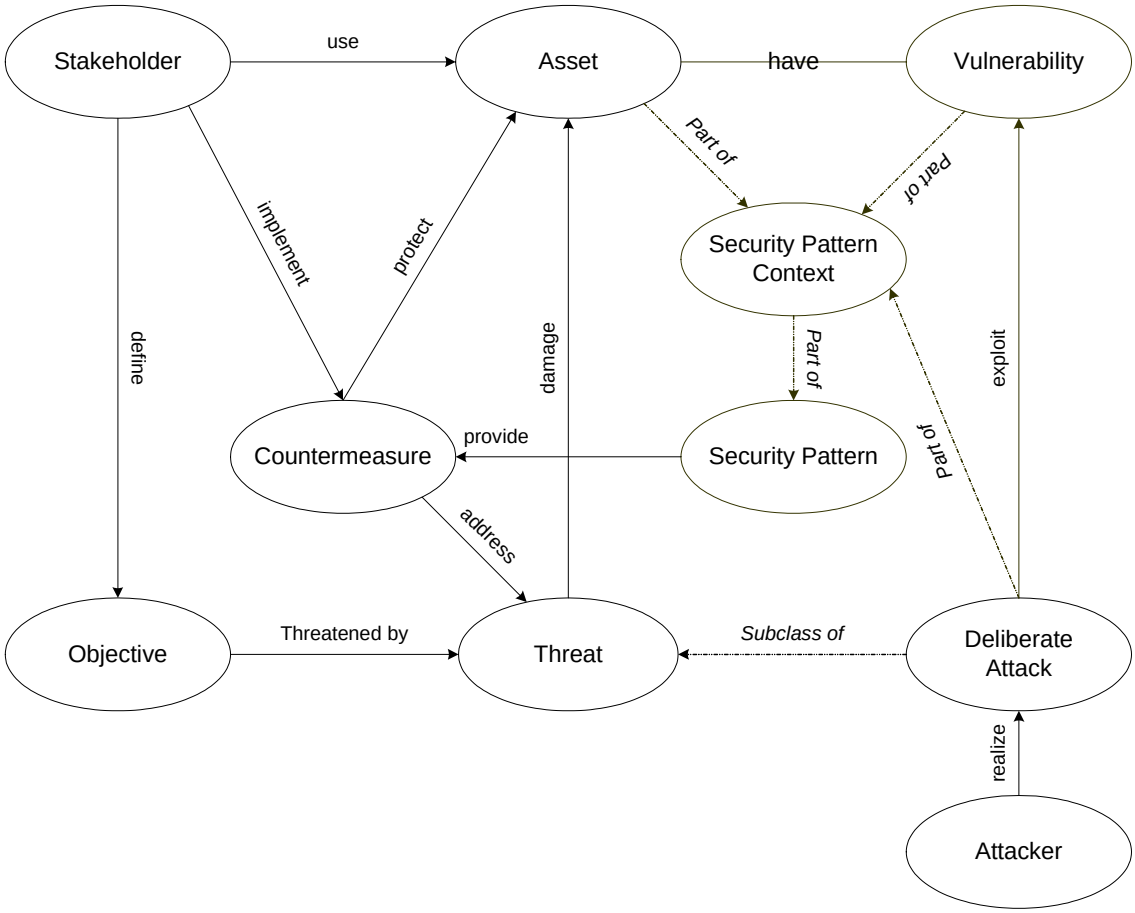

Figure 1: The security ontology (based on Dritsas et al, 2005)

In the depicted Ontology, the concept of a “Security Pattern” is a representation of the security patterns and is connected with the concept of “Countermeasures” with a “*provide*” relationship: each security pattern provides a specific set of countermeasures. In practice each security pattern is matched with a set of countermeasures during the ontology instantiation. A “Security Pattern Context” is defined as a set of “Asset”, “Vulnerability” and “Deliberate Attack” triplets. In this way, one can start from the generic security objectives, find the “Security Pattern Contexts” that match them and, thus, choose specific security patterns. She can then fulfil the high level security requirements and objectives by realizing the respective countermeasures. Thus, the developed ontology comprises a rich source of information regarding the security requirements of the specific application environment and, more importantly, is also a source of information regarding the way several key actors in the software development process view and judge those requirements.

This Ontology provided the basic input for designing an enhanced structure for security patterns, which is more suitable for accommodating the requirements arising in secure applications development. Table 3 presents the security patterns' structure.

Table 3: An enhanced security patterns structure

Element	Description
Name	The name of the specific pattern
Overview	A small description of the pattern
Problem	The problem that the specific patterns addresses
Solution	The mechanism used for confronting the security problem
Requirements	The set of security requirements that the specific pattern satisfies
Assets	The assets related to each application context that the security pattern protects
Threats	The set of threats that the patterns confronts
Vulnerabilities	The set of vulnerabilities that constitute the problem the specific pattern addresses
Related Patterns	Other patterns related to the specific pattern

4. The enhanced security patterns repository

4.1 Patterns comprising the repository

In order to facilitate the use of security patterns in software development process we have designed a security applications repository. The security patterns comprising the repository are structured as described in Table 3. These patterns have different levels of abstraction and they cover technical as well as administrative and procedural aspects of secure applications development as shown in Table 4. In Tables 5 to 7 we present an indicative list of security patterns that belong to the repository.

Table 4: List of patterns included in the security patterns repository and their categories

Pattern Name	Short Description	Pattern category
Authentication	The Authentication pattern incorporates user authentication into the basic operation of an application.	Design Pattern Technical Pattern
Encryption Usage	The Encryption Usage pattern uses specific encryption functions in order to allow security-critical data to be securely stored and/or transmitted over insecure networks.	Design Pattern Technical Pattern
Self-protected Application	The Self-protected Application pattern limits an attacker's ability to discern the internal workings of an application and splits a complex application into two or more simpler components.	Design Pattern Technical Pattern
Patching	Rather than waiting for the system to be compromised before applying patches ("patching reactively"), administrators of software systems should monitor for patches often and apply them proactively.	Architectural Pattern Procedural Pattern
Logging-Auditing	The Log for Audit pattern ties logging to auditing, to ensure that logging is configured with audit in mind and that auditing is understood to be integral to effective logging.	Architectural Pattern Procedural Pattern

Table 5: The “Authentication” Security Pattern

Pattern Name	**Authentication**
Overview	The authentication pattern allows a user to access multiple components-parts-services of a software/application without having to re-authenticate continuously. The specific pattern incorporates user authentication into the basic operation of an application.
Problem	Each user should have access to each component of an application according to its role.
Solution	Provision of specific access control mechanisms in order a user to have access to each component of an application according to its role. Additionally, the authenticated session should keep track of a user’s authenticated identity through the duration of a session so a user accesses multiple protected components of a software application without having to re-authenticate.
Requirements	Confidentiality, Privacy, Accountability, Access Control.
Asset	Source Code of application, User Identities, User Credentials, OS components, Application Components.
Threats	Flawed access control, Identity spoofing, Unauthorized access to various components of an application, Access to files, Access to source code of the software, Denial of Service – Availability problems, OS access.
Vulnerabilities	Source Code Exposure, Configuration Files and Template Files, Debug Functions, Elevation of Privilege, Flawed in ID Access Privilege Check, Missing Authorization Functions, Files Exposure, Weak Authorization.
Related Patterns	Single Sign-On, Authentication Related Patterns.

Table 6: The “Encryption Usage” Security Pattern

Pattern Name	**Encryption Usage**
Overview	The Encryption Usage pattern uses specific encryption functions in order to allow sensitive or otherwise security-critical data to be securely stored on the machine hosting the specific application and/or transmitted over insecure networks.
Problem	The main reason for securing sensitive data on the machine hosting the specific application is because this machine may not be trusted. Many problems might be raised: for example, if attackers manually inspect the contents of an application, they may be able to glean information about the operation of the application that could later be used to compromise the machine etc.
Solution	The Encryption Usage pattern uses encryption techniques to protect data that is stored on the machine and/or transmitted over insecure networks. Using encryption (symmetric or asymmetric) ensures that sensitive data will not be inadvertently revealed. Using message authentication codes or hash functions ensures that the data cannot be tampered with on the client.
Requirements	Integrity, Privacy, Confidentiality, Non Repudiation, Accountability.
Asset	Source code of the application, data manipulated by the application, user identities.
Threats	Intrusion, attacks in the application level, misuse of the application, user errors.
Vulnerabilities	Sensitive data stored in plain text, not enough protection of user accounts, bugs in the application, Insecure networks.
Related Patterns	Encrypted Communication, Secure Remote Authentication.

Table 7: The "Logging – Auditing " Security Pattern

Pattern Name	**Logging-Auditing**
Overview	Applications and components offer a variety of capabilities to log events that are of interest to administrators and other users. If used properly, these logs can help ensure user accountability and provide warning of possible security violations. The Log for Audit pattern ties logging to auditing, to ensure that logging is configured with audit in mind and that auditing is understood to be integral to effective logging.
Problem	As events occur during the life cycle of an application, some events are of particular interest to administrators and other users. Recording specific information about events that have occurred creates records that allow the system to be debugged, monitored for security events, and measured for performance. The term logging means the actual act of writing information about events to some type of permanent storage. The term auditing means actually examining this information and ensuring that all is as expected. There are a number of important problems that logging and auditing solve. They provide evidence of accountability, reliability, performance, and security. Logs provide accountability by enabling verification of an event's occurrence and any users associated with that event. Reliability comes from the logging of errors. Performance analysis can occur when performance data is logged. There are many relevant types of event that must be logged to address security.
Solution	Every major component is responsible for logging events that it considers noteworthy. Some of these will be tagged as security-relevant events, others will not. Each system will typically deliver these events in some non-standard format to permanent storage, using one or more predefined log files. Whenever possible, these log files should be collected centrally and handled consistently. When applications allow the log format to be modified, these features should be used to make logs more consistent.
Requirements	Accountability (Non repudiation)
Asset	Application, OS (in extension)
Threats	Threats related to application (indirect relationship)
Vulnerabilities	-
Related Patterns	-

4.2 Deploying the Security Patterns Repository

In the process of application development, developers have to make decisions regarding possible threats to the applications and the appropriate countermeasures. The use of the proposed repository makes it possible for developers not only to have access to solutions for commonly addressed problems, but also to associate these solutions with specific security requirements, assets, threats and vulnerabilities.

For example, the following questions show how the security patterns repository can facilitate application development regarding security issues:

1. How can the users' passwords be protected against a brute-force dictionary attack?
2. Which vulnerabilities are associated with passwords?
3. Which stakeholders are involved in the implementation of these security patterns (see Q1)?
4. How can a system administrator of an application server enforce accountability?

In each of these situations, developers can use security patterns, which will provide them with information not only for the specific problem and the corresponding solution, but also with information on issues such as the relative threats and vulnerabilities associated with the situation.

5. Conclusions and further research

This paper introduces a new approach for achieving the persisting requirement of incorporating security as early as possible into the software development process. It proposes the use of a security patterns repository. The patterns included in the repository are structured so as to facilitate the fulfillment of security requirements. Furthermore, the proposed enhanced structure of security

patterns provides a solution for overcoming the difficulties and constraints that have been associated with limited use of software patterns.

The security patterns repository developed offers a way to introduce environment specific security requirements to the security patterns selection procedure. It also provides software engineers, who are not security experts, a mechanism to make the appropriate choices regarding security mechanisms and solutions, thus facilitating the development of secure applications. As a next step, this repository will be employed in the development of a security critical application, such as applications that support electronic government services.

6. Acknowledgements

This work was co-funded by 75% from the European Union and 25% from the Greek Government, under the framework of the "EPEAEK: Education and Initial Vocational Training Program—Pythagoras"

References

Buschmann F., Meunier R., Rohnert H., Sommerland P., Stal M. (1996), Pattern-Oriented Software Architecture Volume 1: A System of Patterns, Wiley 1996.

Dritsas S., Gymnopoulos L., Karyda M., Balopoulos T., Kokolakis S., Lambrinoudakis C., Gritzalis S. (2005), "Employing Ontologies for the Development of Security Critical Applications: The Secure e-Poll Paradigm", in *Proceedings of the IFIP I3E International Conference on eBusiness, eCommerce, and eGovernemnt*, Funabashi M., Grzech A. (Eds.), pp.187-201, October 2005, Poznan, Poland, Springer

Gamma E. (2001), Design patterns ten years later. In Broy, M., Denert, E., eds.: Software Pioneers: Contributions to Software Engineering, Springer-Verlag 689–699.

Gamma E., Helm R., Vlissides J., Johnson R., (1993) Design patterns: Abstraction and reuse of object-oriented design. In Nierstrasz, O., ed.: Proceedings ECOOP '93. Volume 707 of LNCS., Springer-Verlag 406–431.

Kienzle D., Elder M., Tyree D., Edwards-Hewitt J., Security Patterns Repository Version 1.0, http://www.scrypt.net/~celer/securitypatterns/template%20and%20tutorial.pdf (accessed on 31/10/2005)

Mekhilef (2003) http://www.knowledgeboard.com/cgi-bin/item.cgi?id=105938&d=pnd, definitions of ontology

Rosengard J., Ursu M. (2004), Ontological Representations of Software Patterns, in the proceedings of KES'04, Lecture Notes in Computer Science, vol. 3215, pp. 31-38, Springer-Verlag, 2004.

Schumacher M. (2003), Security Engineering with Patterns: Origins, Theoretical Models, and New Applications, Paperback, 2003.