Organizing Security Patterns Related to Security and Pattern Recognition Requirements

Michaela Bunke, Rainer Koschke, and Karsten Sohr
Center for Computing Technologies (TZI),
Universität Bremen, Germany
{*mbunke*|*koschke*|*sohr*}*@tzi.de*

***Abstract*—Software security is an emerging area in software development. More and more vulnerabilities are published and highlight the endangerment of systems. Hence, software designers and programmers are increasingly faced with the need to apply security solutions to software systems. Security patterns are best practices to handle recurring security problems. The abundance of documented security patterns calls for meaningful classifications to ease searching and assessing the right pattern for a security problem at hand. Existing classifications for security patterns consider only a small number of patterns and their purpose is often focused on implementation issues. Therefore, we identify missing aspects in existing classifications and the similarities between design and security pattern classifications. Based on that, we introduce two new classification schemes. The first is based on application domains formed by a literature survey on security patterns published in the period of 1997 to mid-2012 to cover the whole bandwidth of existing security patterns. The second is based on a subset of the collected patterns that are concerned with software and combines pattern-recognition needs and security aspects.**

***Keywords*-Security Patterns, Design Patterns.**

I. Introduction

Existing security pattern classifications are often based on a few security patterns. Their scope is often limited to special areas such as implementation patterns. In addition, the heterogeneity of the published patterns in this context is very high. In that context, our paper provides a systematic literature review of published security patterns in the period of 1997 to mid-2012. We propose two new classification schemes. The first summarizes all collected security patterns and organizes them into application domains. The second shows in detail which security and implementation forces security patterns with respect to software have. This classification is an extension of our previous work presented at the International Conferences on Pervasive Patterns and Applications (PATTERNS 2011)[1]. We updated our previous work with new security patterns published till mid 2012, enhanced it with a comparison of design and security-pattern classifications and depict challenges in organizing security patterns.

In the domain of software development, **design patterns** have been proposed as specific solutions for recurring problems in software design [2]. These patterns are also often called **software-design patterns**. Yoder and Barcalow summarized some existing patterns targeting security and introduced the term *security pattern* [3], only three years after Gamma et al. [2] proposed their design patterns. **Security patterns** are best practices aiming at ensuring security [4], [5]. Later on we will use the terminology **software-security patterns** that describe software-related security patterns. These patterns describe security aspects relevant in software design, development, and maintenance.

Existing pattern classifications are mostly based on a small subset of patterns. Their scope is often limited to special areas such as implementation patterns. For instance, Hafiz et al. formed their classification with only 14 security patterns [6], but there exist many more security patterns. Another problem, however, is that information-security experts are rarely development experts [7]. Thus, the usage of existing security patterns and selecting them by way of a classification is a difficult task for non-security professionals who are interested in security aspects.

Therefore, we conducted a systematic literature review and collected the published security patterns in the period of 1997 to mid-2012. We propose a new classification scheme that summarizes 415 security patterns, a much longer list than we found in three surveys [8], [9], [10] and the one by Yoder and Barcalow [3]. Moreover, we shaped this classification scheme towards the selection by application domains, which is relevant for researchers and practitioners who are interested in security patterns.

Retrofitting security aspects into a software system is a difficult task [3]. Accordingly, it would be useful to know which security aspects are already implemented in a software. For instance, Gamma's design patterns can be detected automatically in software systems, but as far as we know, no such approach exists for security patterns [11]. Design and security patterns seem to be very similar except for the security factor, but their concrete similarities and differences are still an open issue in research [11].

Hence, we inspect design and security classifications to determine their similarities and differences to derive possible criteria for a classification that reflect pattern recognition requirements. We use the software-security patterns of our

application-domain classification as a base for this further distinction. Furthermore, our classification is inspired by pattern-recognition needs and combined with the security issues that these patterns solve. For example, this classification can be used by software developers to choose patterns according to particular security requirements. We will use this as a basis for our further research in security-pattern recognition and validation.

The remainder of this paper is structured as follows. An overview of classifications in general is given in Section II. Existing classification approaches for security patterns will be described in Section III. In Section IV, we describe our literature survey and depict challenges in categorizing security patterns in Section V. Afterwards, we will introduce our application-domain classification and specific classification for software-security patterns in Section VI and VII. In Section VIII we will discuss the two presented classifications. Finally, we will conclude and give an outlook in Section IX.

II. Requirements for Classifications

The increasing number of patterns makes it necessary to develop classifications. This section describes requirements for classifications in general and on security patterns in particular.

A classification should be based on systematic methods and techniques to organize a mass of patterns. A classification organizes patterns into groups of patterns that share one or many properties such as the application domain or a particular purpose. The kind of properties that should be used is not fixed and can be customized according to one's needs. A pattern can have more than one specific property. Therefore, it may be included in more than one classification category.

According to Buschmann et al., a pattern classification scheme should meet some basic properties [12]. It must both be simple and easy to learn. This should be supported by using only a few classification criteria to reduce the complexity for users. In addition, a classification should reflect the main properties of a pattern to classify. Last but not least, a classification scheme should provide the possibility to classify new patterns.

Fernandez et al. pointed out that a classification should make the application of patterns much easier along the software life-cycle [13]. Because it is impractical to look at all details of all patterns during pattern selection for the problem at hand, a classification should help to understand the essential nature and value of patterns.

A natural way to classify patterns is to categorize them according to the criteria shown in Figure 1. A simple and intuitive classification can provide one or more of these criteria:

- *Discipline* - categorize patterns according to the discipline when they are applied such as requirements or reverse engineering.
- *Domain* - differentiate patterns by their application domain such as network, embedded systems, or distributed systems.
- *Granularity* - rank patterns depending on the level at which they address a system, e.g., they may address software design or coding patterns.
- *Paradigm* - sort patterns according to paradigms, e.g., programming paradigms such as object-oriented or imperative programming.
- *Purpose* - order patterns by the kind of problem a pattern solves and the point in time it may be applied.
- *Scope* - organize patterns with regard to the characteristic of using them, e.g., class or object representation (see [2]).

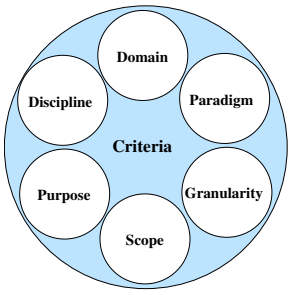

Figure 1. Intuitive classification.

III. Existing Classifications

The presented classification criteria in Section II are simple, but do not always fit for selecting the right pattern for a special purpose because of their generality. Therefore, more specific classification schemata based on one or more criteria have been developed to meet special purposes. Due to the fact that security patterns are formed according to the archetype of design patterns, we will start with classifications for design patterns and continue with existing security-pattern classifications. We will close this section by discussing gaps in security-pattern classifications and whether design-pattern classifications can be used to classify security patterns.

A. Design-Pattern Classifications

Gamma et al. introduced the first classification of design patterns (GoF patterns) [2]. GoF (Gang-of-Four) patterns is an alternative name for the design patterns introduced by Gamma. They classified their patterns based on two criteria: scope and purpose.

As depicted in Figure 2, the "scope" dimension is distinguished by object composition and class inheritance. The purpose dimension is split into *creational*, a *structural* and a *behavioral* criteria. A pattern that is related to an object creation fits into the *creational* criteria. If a pattern is concerned

Scope \ Purpose	Creational	Structural	Behavioral
Class Related	Factory	Adapter (class based)	Interpreter
Object Related	Singleton	Facade	Iterator

Figure 2. GoF classification with a few examples [2].

with compositions or structures that are created by classes or objects, it is called *structural*. The last criterion *behavioral* deals with the way communication or responsibilities are distributed.

Zimmer organized the GoF patterns according to their relationships [14]. He classified the relationships in pairs (X, Y) where X and Y are different design patterns. The relationships are defined as follows:

- *X uses Y in its solution*,
- *X is similar to Y*,
- *X can be combined with Y*,

With these categories, he introduces a new layer structure for pattern classification. According to Figure 3 relationships and structure of patterns are distinguished into three layers:

- *Basic design patterns and techniques*
- *Design patterns for typical software problems*
- *Design patterns specific to an application domain*

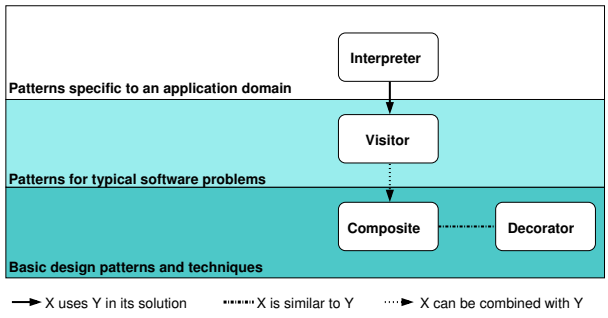

Figure 3. Zimmer's classification with a few examples [14].

Later on, Buschmann et al. presented another organizing approach [12]. They state that all patterns reside on different abstraction layers and it would be more useful to organize them into criteria that express their abstraction level. Therefore, the authors divide their own patterns into three kinds of patterns:

- *Architectural patterns*: specify the fundamental structure of applications.
- *Design patterns*: describe often occurring structures of software-component communication that solve a recurring design problem for a specific context.
- *Idioms*: coding patterns, that is, proven conventions and techniques used during the implementation phase of an application.

Some patterns depend on the technology or domain they are used for and implemented in. These are so-called domain-dependent patterns, e.g., Java Platform, Enterprise Edition (JEE) design patterns. These patterns can be used only in the JEE environment. A system of such patterns has been described by Alur et al. [15]. The authors want to keep the classification simple for their patterns, so they assume "each pattern hovers somewhere between a design pattern and an architectural pattern". These patterns can be classified in the following categories according to their logical tiers (see Figure 4). The *presentation tier* is responsible for creating the presentation used by the client to interact with the user. The *business tier* is responsible for executing the business logic of the application and applies the business logic to the information received from the integration tier. The *integration tier* performs the data-access operations for the application.

Patterns	Tier
Composite View	Presentation
Service Locator	Business
Service Activator	Integration

Figure 4. JEE pattern classification with a few examples.

The design patterns we are discussing exist since 1994. Classifying them is not a highly active research topic in the design-pattern community. An exception is arising new technologies like JEE which require new classifications or the re-evaluation of existing ones. The older ones were not refined further, except for some theoretical abstractions like the one by Hasso and Carlson [16]. They use a complex algebraic structure to classify design patterns.

In 2006, Shi and Olsson identified a lack of classifications for the need of design-pattern recognition [17]. They decided with the hidden agenda of detecting design patterns a new classification approach for the 23 GoF patterns. They suggest a reclassification related to the need of pattern detection by using five categories *language provided*, *structure driven*, *behavior driven*, *domain specific* and *generic concepts*. When a pattern is implemented in some programming language and can be identified by looking at the inheritance hierarchy or specific method names, the pattern is part of the *language provided* category. Patterns that are deeply shaped by their structure and can be identified by their inner-class relationships such as the *Bridge* or *Composite* pattern are *structure driven* patterns. Patterns that have a structure coupled with a specific behavior fit in the category *behavior driven* such as *Singleton* or *State* pattern. Patterns such as *Interpreter*

or *Command* serve domain-specific needs. Detecting such patterns requires domain-specific knowledge. They belong to the *domain specific* category. Patterns in the category *generic concepts* lack a definite structure and behavioral aspects such as the *Memento* pattern.

Their classification allows them to exclude the domain-specific patterns and generic concepts, which cannot be found with common behavioral and structural pattern detections. Moreover, they excluded the language-provided patterns from their detection process because of their easy detection using name matching, too. Patterns that reside in the categories *behavior driven* and *structure driven* were used for design-pattern detection in their tool PINOT. After Shi and Olsson's classification approach, no new design pattern schemata have been developed to the best of our knowledge.

B. Security Pattern Classifications

One of the simplest classifications for security patterns was used by Kienzle et al. [18]. They presented the *structural* and *procedural* criteria for the differentiation of the patterns described in their final report. If a security pattern is concerned with compositions or structures that are implemented in a software product, it is *structural*. If a security pattern improves the process for developing secure software with regard to the organization or management, it is called *procedural*.

Konrad et al. [19] proposed a classification method for security patterns by re-using the classification for design patterns such as *creational*, *structural* and *behavioral* from Gamma et al. [2]. They enhanced their classification by adding further categories such as network, host, and application (see Figure 5). In their work, they considered only the security patterns introduced by Yoder and Barcalow [3].

Purpose / Abstraction Level	Creational	Structural	Behavioral
Application	Session	Check Point Authorization	Limited View Full View with Errors
Host	Session	Check Point Authorization	———
Network	Session	Check Point Authorization	———

Figure 5. The classification by Konrad et al. with a few examples [19].

Schumacher's security patterns book offers a new classification system [20]. The classification is based on Zachman's framework for enterprise architecture [21]. It is presented along two dimensions. One dimension represents different views on the interrogatives "what", "how", "where", "who", "when", and "why". The second dimension shows different information model views such as *business model* or *technology model*. Schumacher et al. enhanced this framework by adding the column *security* to emphasize the security view and to be able to address all model levels. They organized only the patterns contained in the book into their classification.

According to the JEE pattern classification by Alur et al. (see Section III-A), Steel et al. classify their JEE security patterns in a similar way [22]. They separate their patterns in layers that are typical for the development in the JEE domain such as *Web*, *Business*, and *Web Service*, and added a fourth tier that represents the special issue of *identity management* (see Figure 6). This classification is designed only for the special purpose of JEE patterns and does not consider other types of patterns.

Patterns	Tier
Authorization Enforcer	Web
Secure Session Object	Business
Secure Message Router	Web Service
Password Synchronizer	Identity

Figure 6. The classification by Steel et al. with a few examples [22].

Rosado et al. related security requirements to security patterns and classified security patterns into two categories: architectural and design patterns [23].

Hafiz et al. note that simple security-classification concepts are not sufficient to create a partition of security patterns [6]. Their focus is to classify security patterns by their security impact. Their subset of 14 different security patterns is organized by a classification of application context, a Microsoft classification scheme, the CIA [24] and STRIDE model [25]. The acronym STRIDE contains the concepts **S**poofing, **T**ampering, **R**epudiation, **I**nformation disclosure, **D**enial of service, and **E**levation of privilege. Moreover, they proposed a classification based on a tree structure combined with the STRIDE model to join the software and security view in terms of security patterns [6]. The STRIDE model is normally used for threat modeling including identification and prioritization of security vulnerabilities. It is a common tool for security architects who have to prioritize the mitigation effort of security techniques.

VanHilst et al. introduced a multi-dimensional matrix of concerns to classify security patterns [26]. It addresses the problem coverage and pattern classification. Their idea was that each matrix dimension represents a well-defined list of concerns. To classify security patterns, the primary dimension contains concerns of life-cycle activities, such as *domain analysis* or *requirements engineering*. The second dimension differentiates security patterns by their component source type such as *new code*, *legacy*, or *wizard-code*.

Other dimensions may hold types of security responses like prevention or mitigation, but they can also be further customized to a user's need. Their classification was tested by different members of their team, who added six different security patterns to the classification.

Fernandez et al. state that security patterns are architectural patterns [27]. On that account, their approach deals with two classifications that differ in different viewpoints of security patterns. On the one hand, they introduced a classification by a hierarchy of layers and on the other hand, they proposed a classification based on the relationships between patterns by using an automatic relationship extraction and analysis technique. This classification is abstract and regards only a small number of security patterns.

Washizaki et al. point out that the previously introduced classifications have only a few dimensions and do not embrace the relations between patterns [28]. Hence, they introduce a meta model to express the patterns' properties and relations uniformly. The base is an excerpt of the multidimensional classification dimensions presented by VanHilst et al. [26]. They selected the dimensions as follows: *Lifecycle stage*, *Architectural level*, *Concern*, *Domain*, *Type of pattern* and *Constraint*. In addition, they used the three UML standard relationship types *association*, *generalization*, and *aggregation* to model relationships between security patterns, for example, the *Firewall* pattern [20] is the generalization of the *Address Filter Firewall* [29] and the *Application Firewall* [30] pattern.

They also propose two instances for the meta model that represent two points of view, namely pattern-to-pattern relations, represented as a pattern graph, and pattern-to-dimension relations modeled as a dimension graph. They tested their approach with only eight different security patterns that are close to implementation patterns.

C. Classification Similarity

There exists an evolution of design and security pattern classifications with respect to the used classification ideas. We show the influence among security and design-pattern classifications in Figure 7. Some ideas like the purpose of the GoF's design-pattern classification were reused by Konrad et al.'s security pattern classification. Moreover, the criterion *structural* has been adapted by Kienzle et al., whereas the criterion *procedural* and *behavioral* in the GoF classification have different meanings. *Procedural* is used with respect to process patterns for the management or organization of software development in contrast to *behavioral* from the GoF classification where patterns are only software patterns that will be implemented in a software system.

The criteria *architectural* and *design* were proposed by Buschmann's classification scheme et al. [12] and picked up by Rosado et al. [23] and used in conjunction with requirements for a new classification schema.

The three-tier JEE classification [15] and the four-tier

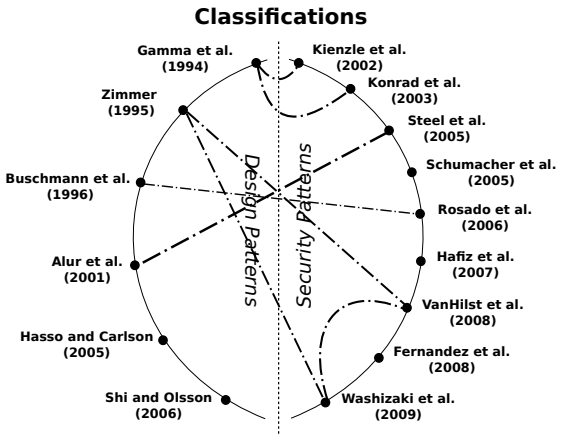

Figure 7. Design and security pattern classification relations – dashed lines between publications highlight alike ideas.

classification for security patterns [22] are very similar, too. They differ only in one additional tier by the security patterns, which deals with identity information. The other three tiers have sometimes different names *Presentation/Web*, *Business/Business* and *Integration/Web Service*, but describe the same criterion (see Figure 4 and 6).

Organizing patterns according to their relationships was introduced by Zimmer and reused by Fernandez et al. and Washizaki et al. , but their relations are different. Zimmer [14] depict a graph with only three predefined types of relationships and Washizaki et al. focus on the UML standard to represent pattern relations like generalization or use relationships [28]. In contrast to that approach, Fernandez et al. use automatically extracted relationships based on the pattern description [27].

D. Classification Distinction

We showed that security-pattern classifications were influenced by design-pattern classifications. All published classifications have one element in common: they take only a small number of patterns into account. On the security-pattern side, the used patterns are often very similar to the patterns first introduced by Yoder and Barcalow [3] and on the design-pattern side the approaches often consider the "core" design patterns described by Gamma et al. [2]. Both subsets of patterns are patterns that will be implemented in a software system. This may lead to the impression that only a handful security and design pattern exist and imply that only programming issues are covered by these patterns. However, Henninger et al. [31] showed in 2007 that there exist more than the few patterns published by Gamma et al. [2] and Buschmann et al. [12]. This statement can also be extended to security patterns, which can also describe enterprise or other security related issues.

Most security pattern classifications are more complex to cover more properties or split purposes or domains into more

dimensions than design patterns. Because of that, we can assume that the security pattern community is aware of the security pattern's heterogeneity, which reflects the additional dimensions in the security-pattern classifications. Design-pattern classifications are often tailored to one group of interest – the software developers. Hence, they are focused on helping to choose the right pattern in design and developing to reach a good code quality and system structure. Due to the fact that the security-pattern audience has more than one group of interest such as security, software development, or enterprise-process design, the security-pattern classifications are built from more heterogeneous criteria than the design-pattern classifications. Yet, not all interests can be covered in one classification. Therefore, more security classifications have been developed till now and developing new ones is still a current topic in the security-pattern community.

A security view is often added to these classifications by using common-threat modeling such as CIA [24] or STRIDE [25]. Adding new views increases the complexity of a classification. A problem, however, is that information security experts are rarely development experts [7]. Because increasing the complexity in security-pattern classifications can make the usage of a classification more difficult for users that have no knowledge or experience with security. It may also lead to difficulties in understanding for other interest groups within the addressed security-pattern audience.

E. Summary

Security patterns related to software can be categorized in a way similar to design patterns. Security patterns that describe other aspects than software-related issues cannot be distinguished by the criteria the aforementioned design-pattern classifications offer, such as "tier", "class related" or "language provided". Therefore, we plan to classify the security patterns in two steps. First, we will look at all security patterns and organize them according to their application domain with respect to their heterogeneity. Secondly, we will focus on software-related patterns and will develop a new classification with existing criteria of design-pattern classifications with respect to software-security patterns.

IV. Collecting Security Patterns

Existing security classifications are limited by the number of chosen security patterns. In addition, existing security pattern surveys are biased by their focus on the same set of security patterns such as the ones of Yoder and Barcalow [3]. The SecurityPatterns website, which provides a short list of security patterns, offers a few more patterns, too, but mixed with articles that describe the application of security patterns [32]. This was not an appropriate position for starting our research activities. Therefore, we decided to conduct a literature survey to provide a proper background for our new classification approach, which should cover the whole range of published security patterns till today. This section describes how our literature survey was conducted systematically following the guidelines by Kitchenham [33], [34].

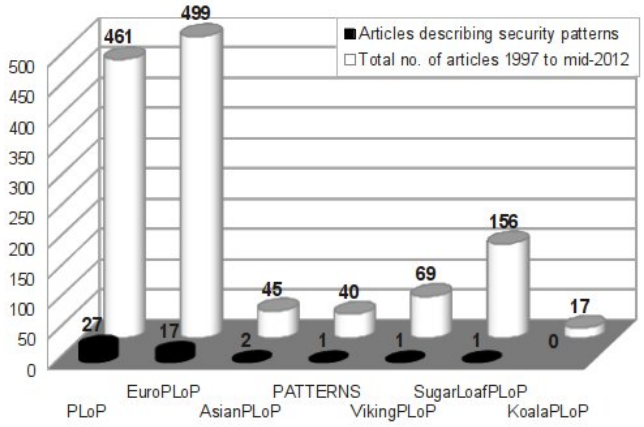

Figure 8. Conferences Involved in the Initial Article Selection.

A. Article Selection and Discovery Process

We started our literature research with the surveys carried out by Laverdiere et al. [8], Heyman et al. [9], and Yoshioka et al. [10]. Their surveys give a good overview of published patterns, but they refer only to often described security patterns. Hence, we also considered common pattern-related conferences (see Table I). and looked for security patterns in the IEEE Digital Library [35] and the ACM Digital Library [36] and considered two security patterns books [20], [22].

The discovery process was split into two parts. One part is the selection of articles published at pattern conferences and the second part is the search for electronic publications.

1) Searching Through Pattern Conferences: The aforementioned pattern conferences (see Table I) of the years 1997 to mid-2012 were skimmed for several keywords, such as cryptographic, security, software, or secure. At first, we picked out all publications that contain these keywords. In this initial selection phase, we found 1268 articles (see Figure 8). Secondly, we read the abstract if it described the presentation of a security pattern and made a note of the authors, publication year, and title. Thereafter, we read the publications not filtered out previously to verify that they describe security patterns. In this step, we enhanced our list with each identified pattern for further readings. Finally, we scanned the publication references and collected referenced publications containing the aforementioned keywords. We stopped the step of cross-reference scanning when we did not find new publications containing the keywords we searched for.

2) Searching Through Electronic Publications: On searching for other electronic publications we used the two

Acronym	Description
AsianPLoP	Asian Conference on Pattern Languages of Programs
EuroPLoP	European Conference on Pattern Languages of Programs
KoalaPLoP	Australian Conference on Pattern Languages of Programs
PATTERNS	International Conferences on Pervasive Patterns and Applications
PLoP	Conference on Pattern Languages of Programs
SugarLoafPLoP	American Conference on Pattern Languages of Programming
VikingPlop	The Nordic Conference on Pattern Languages of Programs
GRID-STP	International Workshop on Security, Trust and Privacy in Grid Systems
ICOMP	International Conference on Internet Computing
IFIP WC 11.3	Working Conference on Data and Applications Security
SECURWARE	International Conference on Emerging Security Information, Systems and Technologies

Table I
CONFERENCES INVOLVED IN THE INITIAL ARTICLE SELECTION. A GREY BACKGROUND INDICATES THE DISCOVERY BY CROSS-REFERENCES.

digital libraries provided by IEEE and ACM [35], [36]. Both offer an extensive database search for published publications. First of all, we used the simple search to find publications that contain the aforementioned keywords. Due to the fact that the number of results was very high and too unspecific, we used the advanced search field provided by the websites to obtain more localized results. There, we concatenate the following options with "and" and limited them to get better results:

- The year of publication date has been limited from the year of the first published security patterns 1997 to mid-2012.
- The full text must contain the word "pattern".
- The title must contain one of the aforementioned keywords.

Unfortunately, these restrictions still provided many unwanted results. Therefore, we skimmed the result list from top to bottom – where the search engines of the digital libraries provided the ordering based on relevance – and discontinued if we read more than ten papers that do not deal with security patterns in their abstract.

Further, we skimmed the collected patterns like the aforementioned conference publications. We verified that they describe one or more security patterns and then collected their cross references.

B. Summary

We identified 67 different publications describing security patterns, including books, journals, proceedings, and technical reports. Most of them were found by looking at the Hillside Group [37] pattern conferences such as PLoP and EuroPLoP (see Figure 9 and Table I) and books. Another publication type containing many security patterns were technical reports discovered by cross references. New conferences that have only a few security-pattern publications are also discovered by cross references (see Table I).

The search at the ACM and IEEE Digital Library produced many false-positive articles that were at a closer look no security pattern descriptions, but deal with them in other ways like discussing secure software design in practice [38].

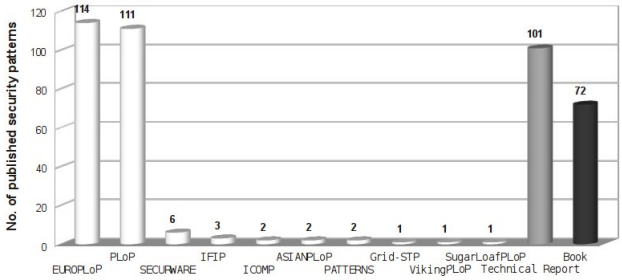

Figure 9. Distribution of security patterns across different venues; white bars denote conferences, the grey bar technical reports, and the black bar books.

Some publications describe more than one pattern. In total, we got 415 security patterns. This list identified that some of these patterns have been described more than once. Hence, we filtered out duplicates and reduced the number of patterns to 364. These duplicates were identified by the use of similar names and then comparing their descriptions. Because of the abundance of patterns, we were not able to check in depth whether two patterns with different names relate to the same concept. This was also forced by the nonuniform descriptions of the patterns, which will be discussed in the next section.

V. CHALLENGES IN CATEGORIZING SECURITY PATTERNS

Identifying duplicates is not the only challenge in categorizing security patterns. We agree with Yoshioka et al. [10] and Heyman et al. [9] that the abstract description of patterns is another challenge. Because the quality of the security-patterns descriptions may influence the categorizing outcome, we spent some time in inspecting the description forms.

A. Description Form Inspection

Various descriptive models for security patterns exist within the security-pattern community [39]. The POSA[1] model described by Buschmann et al. is said to be frequently used to describe the context and usage of security patterns [12]. Yet, during our literature review we observed that the patterns are often described in custom styles and do not follow strictly the POSA model.

Therefore, we gather all used description aspects in the collected security-pattern publications to review how security patterns are described formally. We assume that description aspects that are often used in several pattern descriptions provide the best clues for selecting and organizing security patterns.

B. Section Assessment/Examination

There exist 63 different sections or aspects such as *Problem*, *Intent* or *Known Use* that are used in the 67 collected security pattern publications. The heterogeneity in naming the aspects is very high and no mapping between the different names exists like the one by Henninger et al. between the POSA and GoF descriptions [31]. For this reason, we identified significant sections which can be used to get a first impression on security patterns and make patterns comparable (see Table II).

Context is a frequently used description aspect with 49 of 67 hits, but the context description is often very short. In some publications, it consists only of one or two sentences (see [40] or [41]). This circumstance makes it hard to obtain sufficient knowledge or even an idea of what the pattern is about. Similar findings were made by Laverdiere et al. for the naming and the section *Intent* in security-pattern descriptions [8].

The **Problem** aspect occurs in about 84 percent of the publications. In many cases, these problem descriptions are abstract or describe a simplified problem for the security pattern (e.g., [40]). Therefore, this description aspect is less applicable to categorizing security patterns for an application domain but suitable to gather the security aspects it addresses.

The **Related Patterns** aspect requires a good knowledge of other security patterns and their application domains to be used for a distinction. This also applies to the **Consequences** aspect where additional knowledge is required to be able to relate to application consequences in the security area. Hence, these sections cannot be recommended for novices in security to accomplish a pattern distinction.

A **Known Use** aspect depicts where a pattern can be found in real life. This section often labels software or software parts like an application login screen, UNIX telnet or Linux as operating system software. The given keywords

[1]POSA is the acronym of the design-pattern book series "Pattern-Oriented Software Architecture" written by Buschmann et al. [12].

Sections	Used by # Publications
Solution	58
Problem	56
Related Patterns	50
Consequences	50
Context	49
Known Use	46
Example	33
Forces	27
Example Resolved	25
Structure	25
Implementation	23
Dynamics	21
Intent	22
Also Known As	10
Motivation	10
Participants	8
Applicability	7
Variants	7
See also	7
Collaboration	5
Sample Code	5
Alias	4
Impairments	3
Resulting Context	3
Abstract	3
Benefits	2
Features	2
Properties	2
Preconditions	2
Resultant Context	2
Running Example	2
Alternatives	1
Class-Diagram	1
Classification	1
Conflicts	1
Contradictions	1
Dependencies	1
Design Issues	1
Example Instances	1
Hardware/Software	1
Implementation Factors	1
Implementation Issues	1
Implementation Example	1
Issues	1
Labels	1
Liabilities	1
Non-Security Known-Use	1
Non Software Example	1
Other Example	1
Participants & Responsibilities	1
Rationale	1
Reality Checks	1
Relationships	1
Resolved Example	1
Resulting Context	1
Security Factors and Risks	1
Security Objectives	1
Solution Example	1
Solution Implementation	1
Social Dependencies	1
Specific Context	1
Strategies	1
Trade-Offs	1

Table II
ASPECTS WHICH ARE USED BY SECURITY-PATTERN DESCRIPTION FORMS. THE OFTEN USED ASPECTS ARE HIGHLIGHTED WITH A GREY BACKGROUND.

and explanations within this section give a good impression to which domain this pattern can be applied.

A **Solution** aspect is used in about 87 percent of the 67 publications. The described solution in the collected publications is frequently used and provides often a good depiction of the security pattern and the problem it solves. Moreover, this section often describes which security aspects are covered by the pattern and how this could be implemented by software-security patterns. If the description does not provide a solution aspect, it can hardly be considered a pattern description. One may argue that the GoF and POSA description templates neither provide a solution aspect, but they describe this aspect in a refined manner using the description aspects *Structure*, *Implementation* and *Running Example*. This heterogeneity in the form of descriptions is one aspect that we have to deal with and will be discussed in the following section.

C. Challenges for our Classification Approach

Besides the high variation in the pattern description quality, we note that not all often occurring aspects are equally useful to get a quick access to a pattern's goal and application domain.

If one does not have the time to read a whole pattern description or has a lack of sufficient security knowledge to understand the described pattern, we propose to look at first at the *Known Use* aspect to get an idea of the pattern's application domain. On account of the good depiction of the security pattern and the problem it solves in the *Solution* aspect, we can recommend in a second step to look at this section, if there exists no *Known Use* section or if the containing information is not satisfactory. In addition, the *Solution* aspect gives hints on how the pattern can be implemented in software or used for end users or enterprise processes.

With this strategy, approximately 80 percent of the patterns can be sufficiently understood. The remaining 20 percent can only be organized by reading the full pattern description because of their insufficient description structure in comparison to the majority of security publication. So we decided to read the whole pattern description for each classification because of the high description heterogeneity and high varying description quality.

During the pattern description-form examination process we observed that some description-form aspects are filled in an insufficient way. An example is the publication by Yskout et al. where many aspects in the pattern description form exist, but many of them are filled with one or two words or with a few sentences [42]. Due to the fact that such an imprecise description leaves much room for interpretation and imagination about what the pattern describes, it increases the difficulty in the distinction process for a new classification. It may also compromise the correctness of distinction.

Many security-pattern description forms follow in some aspects the POSA Template, but they are compounded by different terminology like *Problem* and *Motivation* or *See Also* and *Related Patterns*, which describe the same issue in the publications. A uniform form of description is desirable. Research should aim at improving the quality of security-pattern descriptions. Initial work along this line has been done but only for a small subset of patterns [9], [43].

VI. Application Domain Classification

The new classification unifies the existing patterns into a common scheme. In addition, not every task needs information about attack surfaces or vulnerability classification properties like STRIDE or other facets that are introduced in Section III. On that account, we omit specialized criteria like STRIDE and focus on universal differences among the security patterns. With this in mind, we develop a new classification with a more general perspective based on a domain criterion (see Section II) and the security patterns we collected in our systematic literature review (see Section IV).

A. Organizing by Application Domain

To derive our classification, we first skimmed over the data and collected keywords for the security patterns such as user, password, operating system, enterprise or process. These keywords were gathered by information we found in the pattern descriptions.

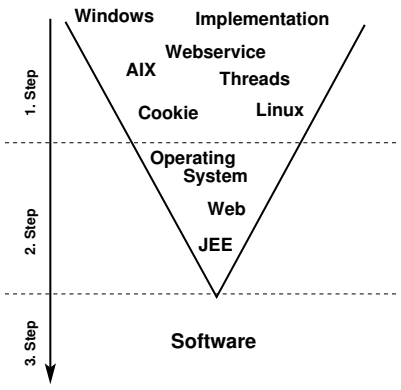

Figure 10. Proceeding steps in our classification model.

In the next iteration, we went through the pattern list and extracted keywords for the patterns. On further reading, these keywords were unified into common groups. For instance, we united the keywords *AIX*, *Linux* and *Preforking* to the group *Operating System*. The result contains a mixture of purpose and domain criterion. We formed 13 different groups this way. To further simplify the classification along the lines described in Section II, these keywords were further condensed to form an application-domain based distinction, which is easy to understand and intuitively applicable (see

Figure 10). Finally, Figure 11 depicts the five target application domains that were discovered: *Enterprise*, *Software*, *Cryptographic*, *User*, and *Network*. They are described in the following in more detail.

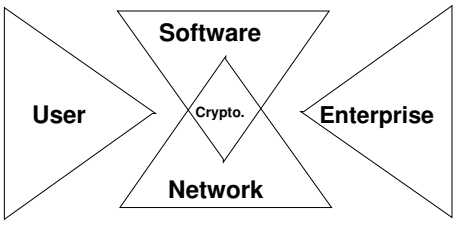

Figure 11. Application-domain based classification.

B. The Application-Domain Criteria

Enterprise-security patterns deal with aspects that are important for enterprises to ensure security in several enterprise segments such as third-party communication with suppliers. This means security in processes, physical authentication to several areas, risk mining or securing communication in internal and external businesses. A good example of this pattern type is the *Manage Risk* pattern introduced by Elsinga and Hofman [44]. The problem addressed by this pattern is as follows: "What is the right (combination of) paradigm(s) to formulate the corporate security strategy in order to select and implement the appropriate set of security safeguards?" The pattern suggests to instruct people and units to pay attention on known and unknown risks to develop prevention and roll-back strategies.

Network-security patterns address network infrastructures and their ideal composition. For instance, the *Packet Filter Firewall* pattern describes how to shield an internal network from Internet attacks just by tunneling the communication traffic through a single controllable instance [20] and the *Virtual Private Network* pattern [39] depicts how secure connections over public networks such as the Internet can be established. The point-to-point tunneling protocol (PPTP) is a specific implementation of this pattern [97].

User-security patterns are focused on user behavior or their awareness of security issues, for example, the *Password Lock Box* pattern, which encourages the user to protect master passwords with the highest level of security [52]. It stresses the significance of protecting master password files and depicts situations where such a file can be useful. The *Keep It Secret* pattern [52] highlights that published or publicly known passwords pose a potential danger to be misused by attackers. To minimize this effect, one should keep a password secret or use *Password Salt* (another security pattern) to vary the password [52]. Another pattern in this domain describes how one can configure the web browser to control how and when cookies are set and used [54].

Software-security patterns describe mostly how to structure parts of software to ensure security requirements. Sometimes they also describe a specific behavior or way to manage or control a data flow in a secure way. On one hand, patterns in this domain can be very specific like JEE patterns, which can be applied only to Java enterprise applications [22]. An example is the *Container Managed Security* pattern [22], which is a standard way to enforce authentication and authorization in a JEE application so that no special hard-coded security policies are necessary. On the other hand, patterns in this domain can be more general like the *Single Access Point* pattern, which models a kind of login structure that can be found in several software systems like UNIX, ICQ or Twitter [3]. Patterns of this application domain can also be called *Security Design Patterns* along the lines of the GoF design patterns, which also focus on software.

Cryptographic security patterns depict secure communication between two applications over a network. They are often described abstractly. Therefore, it is not clear whether these patterns reside in the *Network* or *Software* domain. Their implementation or application is possible in both domains. On that account, we view them as a part of network and software in our classification (see Figure 11). An example is the *Sender Authentication* pattern. It presents the problem and solution how to guarantee that a received message has been sent by a person one expected [40]. Obviously, such a pattern can be applied at network level (level 3 and 4) or at application level, and depending on that, it resides on the *Network* or *Software* application domain.

The aforementioned classifications in Section III cover only parts of the fields we discovered. The *Network* domain is partly touched by the classification of Konrad et al. [19]. Schumacher et al. [20] factor *Enterprise* requirements customizable with viewpoints in their classification, but they do not distinguish other domains as our approach does. The domains *User* and *Cryptographic* are not mentioned in the existing classification approaches, although they represent approximately one sixth of the patterns (see Table III).

VII. Merging Pattern Recognition and Security Needs

The application-domain classification scheme can be tailored further to practical or research interests by employing, for example, viewpoints as recommended by Fernandez et al. [27]. For software engineering in particular, applicable patterns are located in the category *Software*, which can be further divided into specific purposes such as pattern detection by using the existing pattern classifications by Shi and Olsson [17]. Developing new viewpoints or finer grained classifications to cover new needs in terms of special purposes for one of the application domains is also conceivable.

An interesting issue for us is the effect of implemented

Application Domain	Publications Describing Security Patterns	Total no. of Security Patterns
Enterprise	[20], [45], [46], [47], [44], [48], [49], [50], [51]	86
User	[52], [53], [54]	24
Cryptographic	[40], [55], [56], [57], [58]	37
Network	[20], [29], [39], [30], [41], [59], [60], [61], [62], [63], [64], [65], [66], [67], [68], [69], [70], [71], [72], [73]	56
Software	[3], [18], [20], [22], [42], [47], [50], [54], [55], [57], [59], [63], [64], [67], [74], [75], [76], [77], [78], [79], [80], [81], [82], [83], [84], [85], [86], [87], [88], [89], [90], [91], [92], [93], [94], [95], [96]	161

Table III
PUBLICATIONS AND NUMBER OF SECURITY PATTERNS PER APPLICATION DOMAIN

security aspects in software. The usage of security patterns can harden and protect a software against common threats [4], [5]. Halkidis et al. [98] showed that one can determine architectural risks for a software systems through the fact of usage or non-usage of security patterns. Therefore, we assume that software which is designed using security patterns is more secure, and if we are able to detect these patterns, we can rate the degree of security of a software system by the usage of security patterns. This is comparable to software design patterns, which imply a higher code quality when used appropriately.

A. Classification

Our approach is motivated by searching for security patterns in software to be able to determine the built-in security mechanisms of a software system. The application-domain classification scheme indicates which security patterns are relevant for designing software. Due to the fact that this classification scheme is very general, we decided to tailor the patterns in the *Software* domain further to our research goal. We chose two dimensions for our classification to show the pattern's security impact and its purpose in terms of software development. The first dimension represents the pattern-recognition aspects and the second common security aspects (see Figure 12). All classified software-security patterns in detail can be found in the Appendix. Our classification dimensions will be described in depth in the following sections.

1) Pattern-Recognition Aspects: Early approaches to design-pattern detection date back to the year 1996 [99]. There exist several specialized approaches of pattern recognition that use different aspects of patterns for their detection, such as structural, behavioral aspects or software metrics. The common matching techniques are all based on structural and/or behavioral aspects, e.g., [17], [99], [100], [101]. Patterns of these aspects need different information and analyses to be automatically detected in a software system. Having a distinction for these aspects is very helpful to define which analyses and what kinds of information are needed to find a specific pattern.

Some existing security-pattern classifications depicted in Section III are formed on ideas from formerly published design-pattern classifications. Thus, they imply that security patterns are like design patterns. We gave another picture of the security-pattern landscape with our application-domain classification. Due to the fact that we extracted software-security patterns of the whole set of security patterns, we can use some of the design-pattern classification criteria for our needs.

Representing information relevant to pattern recognition, we choose some criteria of the design-pattern classification of Shi and Olsson [17]. **Structural** software-security patterns are characterized by their particular class structure. This structure can be realized by inter-class relationships like inheritance, association or delegation relationships (see also [17]). The *Single Access Point* pattern [3], [20] is such a structure-driven pattern. It provides a single access to a protected system. The focus, is on the pattern's structure in a software described through static relations rather than its behavior at runtime. **Behavioral** software-security patterns are primarily designed from a behavioral point of view. They can be easier described and found by their typical behavior than their structure (see also [17]). An example of such a pattern is the *Secure Logger* [22]. It is a class that manages the logging of data in a secure and centralized manner. **Generic Concept** security patterns describe very general solutions for security problems. Unfortunately, many of the patterns in this category do not provide implementation details, UML diagrams or other information, such as example code snippets, that can be used to distinguish between a structural or behavioral character of the pattern. An example is the *Password Authentication* pattern [18], which describes

the secure management of passwords while designing a user login.

Shi and Olsson [17] also provide the criteria *Domain Specific* and *Language/Framework provided* but they cannot be directly used for our security-pattern classification. Due to the fact that security patterns are mostly written in an abstract way it is not possible to classify a pattern according to these two criteria. A similar analysis has to be made for the language-provided patterns which manifest through a framework-specific structure or method names for this type of patterns.

2) Security Aspects: Developers tend to view IT security in terms of software requirements rather than taking the perspective of an attacker. Software patterns are usually chosen by developers with a particular goal in mind. For this reason, we employ general security goals for our pattern classification and not the STRIDE model, which focuses on the attacker's perspective. In the area of IT security, the most common goals are confidentiality, integrity, and availability of data [102]. **Confidentiality** guarantees the secrecy of data, whereas **integrity** makes sure that data are not modified in an unauthorized way. The *Secure Visitor* pattern fulfills the latter aspect. Nodes can only be accessed by a *Secure Visitor* who prevents unwanted access and unauthorized modifications of nodes in hierarchically structured data. **Availability** means that data/services are accessible. The *Keep Session Data in Client* pattern [92] provides the accessibility to a website if the connection between client and server is interrupted for a short time. Sometimes, **non-repudiation** (proving an action to a neutral and trustworthy third party) and **accountability** (logging certain actions for audits) are also of interest. One pattern example of these two aspects is the *Audit Interceptor* pattern [22]. It intercepts audit requests and responses to and from the business tier in JEE applications and logs them in an appropriate way. In addition, the identification of principals (e.g., users, machines, and processes), called "**authentication**", is important as well as **access control**, which determines which principal may access which data. Both aspects are used in the *Secure Visitor* pattern [20] where the visitor has to verify a user's credentials and check it against the access control rules for modification. As a consequence, our classification considers confidentiality, integrity, availability, non-repudiation, accounting, authentication, and access control in the second dimension.

B. Results

We organized all software-security patterns that we described in Section III according to the aforementioned aspects (see Appendix for details). We detected that the software-security patterns often describe integrity and confidentiality problems (83 and 62 times, respectively). Authentication and access control issues are often used, too. In total, 58 and 53 patterns per criterion can be found. Lesser attention in the software-security patterns have the security aspects availability, accountability, and non-repudiation with 22, thirteen, and two patterns, respectively, that deal with these problems. As depicted in Figure 12, we found no structural-driven pattern that covers accountability and no generic-concept pattern that handles non-repudiation aspects. All other security criteria are matched by a software-security pattern.

We detected 58 *behavioral* and 37 *structural* characterized software-security patterns. The majority of the patterns are *generic concepts*. *Generic concept* patterns cannot be directly used for a pattern recognition approach. Most of these patterns do not provide implementation information like example code that could enable a distinction into structural or behavioral. A distinction can be made if further inspections on the real usage of these pattern in software systems have been conducted and concrete implementations can be found and assessed. With this additional investigation, it may be possible to detect such patterns in the future and give designers a better idea of how to design and implement these patterns in a software system. We also expect that for some of the generic concept patterns like *Red Team The Design*, we will find no implementations in software.

VIII. Discussion

The presented application-domain classification scheme fulfills the requirements of classifications in terms of expandability, intuitive use, and is applicable for security laymen. This approach can be expanded by repeating the proceeding steps described in Section VI-A for new patterns if new application domains for security patterns emerge. The intuitive use and the applicability for security laymen is supported by the usage of only one criterion – selection by domain– which is easy to decide for a user. We suppose that a user knows in what domain she will work with security patterns, e.g., an enterprise process designer may select *Enterprise* as her application domain.

We expect that the application-domain classification helps other researchers and practitioners with specific application goals focusing on security patterns. A possible use case for this classification is, e.g., when an enterprise process architect is looking for a best practice to administer threats and risks for his enterprise. Then she can have a look at all enterprise-related patterns listed in the publications of Table III and will find patterns like *Risk Determination* and *Threat Assessment* [20] to solve her problems.

Furthermore, the second classification can help software designers to choose the right security patterns for their software system according to specific security requirements. This classification gives also a detailed overview which patterns can be used in general with respect to software related issues. Moreover, it allows one to select a software-security pattern according to its security attributes for the software design phase or determining security features for a

Security \ Recognition	Structural	Behavioral	Generic Concept
Accountability	———	Audit Interceptor Secure Logger	Password Authentication
Authentication	Single Access Point Subject Description	Secure Visitor	Password Authentication
Access Control	Check Point Subject Description	Secure Visitor	Keep Session Data In Client
Availability	Partitioned Application	Secure Preforking	Keep Session Data In Client
Confidentiality	Partitioned Application	Secure Visitor	Password Authentication
Integrity	Check Point	Secure Preforking Secure Visitor	Password Authentication
Non–Repudiation	Subject Description	Audit Interceptor Secure Logger	———

Figure 12. Our software security-pattern classification with a few examples.

security assessment.

Besides the open issue – which concrete similarities and differences design and security pattern have [11] – we identified two additional gaps in research. One is that additional work must be done to define a uniform description for security patterns to increase the description quality. As mentioned in Section V, some work has been done in this area but as we showed, the heterogeneity even by newer pattern publications is still very high. On account of this, it is desirable to have all security patterns available from a single source and presented in a uniform format like other existing open databases for design patterns (e.g., [103], [104]).

Another open issue is that we found no structural patterns with accountability aspects and no general concept patterns with non-repudiation properties. The absence of these aspects indicates a gap in the software-security pattern landscape.

Additional investigations are necessary for all software-security patterns not only *generic concept* patterns. For our classification, we were able to decide whether a pattern falls into the category of structural or behavioral patterns, but the pattern descriptions are often not sufficient and exact enough to use existing pattern recognitions out of the box for their detection. It remains a high variability in their possible implementations.

IX. Conclusion and Outlook

In this paper, we presented our systematic literature review on security patterns, a comparison of design and security-pattern classifications, discussed challenges in classifying security patterns, and introduced two new classification schemes.

The first classification scheme embraces 364 published security patterns and exceeds in numbers existing classifications by far. The second classification unites the focus of pattern recognition and security aspects. It classifies 161 software-security patterns that we obtained in the first organization process.

This classification will support our future research by the determined pattern characteristics and the indicated open issues (see Section VIII). In particular, we plan to detect and validate software-security patterns implemented in code. Automatically detected security patterns can support security and risk assessments and help in reengineering existing software systems.

X. Acknowledgments

This work was supported by the German Federal Ministry of Education and Research (BMBF) under the grant 01IS10015B (ASKS project).

References

[1] M. Bunke, R. Koschke, and K. Sohr, “Application-domain classification for security patterns,” in *Proceedings of the International Conferences on Pervasive Patterns and Applications*, IARIA Conferences. XPS (Xpert Publishing Services), 2011, pp. 138–143.

[2] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Object-Oriented Software*. Addison Wesley, 1994.

[3] J. Yoder and J. Barcalow, “Architectural patterns for enabling application security,” in *Proceedings of the Conference on Pattern Languages of Programs*, Monticello/IL, 1997, pp. 1–31, last access: 23.06.2012. [Online]. Available: http://hillside.net/plop/plop97/Proceedings/yoder.pdf

[4] S. Haldikis, A. Chatzigeorigou, and G. Stephanides, “A practical evaluation of security patterns,” in *Proceedings of the International Conference on Artificial Intelligence and Digital Communications*, Aug. 2006, pp. 1–8, last access: 23.06.2012. [Online]. Available: http://inf.ucv.ro/~aidc/proceedings/2006/5%20shalkidis.pdf

[5] M. Hafiz and R. E. Johnson, "Evolution of the mta architecture: the impact of security," *Software—Practice and Experience, Wiley*, vol. 38, no. 15, pp. 1569–1599, 2008.

[6] M. Hafiz, P. Adamczyk, and R. E. Johnson, "Organizing security patterns," *IEEE Software*, vol. 24, pp. 52–60, 2007.

[7] K. R. van Wyk and G. McGraw, "Bridging the gap between software development and information security," *Security Privacy, IEEE*, vol. 3, no. 5, pp. 75–79, Sep. 2005.

[8] M. Laverdiere, A. Mourad, A. Hanna, and M. Debbabi, "Security Design Patterns: Survey and Evaluation," *IEEE Canadian Conference on Electrical and Computer Engineering*, pp. 1605–1608, 2006.

[9] T. Heyman, K. Yskout, R. Scandariato, and W. Joosen, "An analysis of the security patterns landscape," in *International Workshop on Software Engineering for Secure Systems*. Washington, DC, USA: IEEE Computer Society, 2007, p. 3.

[10] N. Yoshioka, H. Washizaki, and K. Maruyma, "A survey on security patterns," *Progress in Informatics*, vol. 5, pp. 35–47, 2008.

[11] M. VanHilst and E. B. Fernandez, "Reverse engineering to detect security patterns in code," in *Proceedings of the International Workshop on Software Patterns and Quality*. Information Processing Society of Japan, Dec. 2007, pp. 25–30.

[12] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*. Chichester, UK: Wiley, 1996.

[13] E. B. Fernandez, N. Yoshioka, and H. Washizaki, "Using security patterns to build secure systems," in *Proceedings of the International Workshop on Software Patterns and Quality*. Information Processing Society of Japan, 2007, pp. 47–48.

[14] W. Zimmer, *Pattern languages of program design*. New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 1995, ch. Relationships between design patterns, pp. 345–364.

[15] D. Alur, D. Malks, and J. Crupi, *Core J2EE Patterns: Best Practices and Design Strategies*. Upper Saddle River, NJ, USA: Prentice Hall PTR, 2001.

[16] S. Hasso and C. Carlson, "A theoretically-based process for organizing design patterns," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–22, last access: 23.06.2012. [Online]. Available: http://hillside.net/plop/2005/proceedings/PLoP2005_shasso0_3.pdf

[17] N. Shi and R. A. Olsson, "Reverse engineering of design patterns from java source code," in *Automated Software Engineering*. Los Alamitos, CA, USA: IEEE Computer Society, 2006, pp. 123–134.

[18] D. M. Kienzle, M. C. Elder, D. Tyree, and J. Edwards-Hewitt, "Security patterns repository, version 1.0," 2003, last access: 23.06.2012. [Online]. Available: http://www.scrypt.net/~celer/securitypatterns/repository.pdf

[19] S. Konrad, B. H. Cheng, L. A. Campbell, and R. Wassermann, "Using security patterns to model and analyze security requirements," in *International Workshop on Requirements for High Assurance Systems*, 2003, pp. 13–22.

[20] M. Schumacher, E. B. Fernandez, D. Hybertson, and F. Buschmann, *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, 2005.

[21] "The zachmann framework for enterprise architecture," 2012, last access: 23.06.2012. [Online]. Available: http://zachmaninternational.com/2/Zachman_Framework.asp

[22] C. Steel, R. Nagappan, and R. Lai, *Core Security Patterns: Best Practices and Strategies for J2EE(TM), Web Services, and Identity Management*. Prentice Hall International, 2005.

[23] D. G. Rosado, C. Gutiérrez, E. Fernández-Medina, and M. Piattini, "Security patterns related to security requirements," in *Proceedings of the International Workshop on Security in Information Systems*, 2006, pp. 163–173.

[24] Commission of European Communities, "Information technology security evaluation criteria, ver. 1.2," 1991, last access: 23.06.2012. [Online]. Available: https://www.bsi.bund.de/cae/servlet/contentblob/471346/publicationFile/30220/itsec-en_pdf.pdf

[25] F. Swiderski and W. Snyder, *Threat Modeling (Microsoft Professional)*. Microsoft Press, 2004.

[26] M. VanHilst, E. B. Fernandez, and F. A. Braz, "A multi-dimensional classification for users of security patterns," in *Proceedings of the International Workshop on Security in Information Systems*, 2008, pp. 89–98.

[27] E. B. Fernandez, H. Washizaki, N. Yoshioka, A. Kubo, and Y. Fukazawa, "Classifying security patterns," in *Proceedings of the Asian-Pacific Web Conference*, Apr. 2008, pp. 342–347.

[28] H. Washizaki, E. B. Fernandez, K. Maruyama, A. Kubo, and N. Yoshioka, "Improving the classification of security patterns," *Database and Expert Systems Applications*, pp. 165–170, 2009.

[29] E. B. Fernandez, M. M. Larrondo-petrie, N. Seliya, N. Delessy, and A. Herzberg, "A pattern language for firewalls," in *Proceedings of the Conference on Pattern Languages of Programs*, Sep. 2003, pp. 1–13, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2003/Papers/Fernandez-firewalls.pdf

[30] S. R. Nelly Delessy-Gassant, Eduardo B. Fernandez and M. M. Larrondo-Petrie, "Patterns for application firewalls," in *Proceedings of the Conference on Pattern Languages of Programs*, 2004, pp. 1–19, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2004/papers/ndelessygassant0/PLoP2004_ndelessygassant0_0.doc

[31] S. Henninger and V. Corrêa, "Software pattern communities: Current practices and challenges," in *Proceedings of the Conference on Pattern Languages of Programs*, ser. PLOP '07. New York, NY, USA: ACM, 2007, pp. 14:1–14:19.

[32] "Securitypatterns.org," 2012, last access: 23.06.2012. [Online]. Available: http://www.securitypatterns.org/

[33] B. Kitchenham, "Procedures for performing systematic reviews," Keele University, Keele, UK, Technical Report TR/SE-0401, 2004.

[34] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Keele University, Keele, UK, Technical Report EBSE-2007-001, 2007.

[35] IEEE, "IEEE Digital Library," 2012, last access: 23.06.2012. [Online]. Available: http://www.computer.org/portal/

[36] ACM, "ACM Digital Library," 2012, last access: 23.06.2012. [Online]. Available: http://portal.acm.org/

[37] The Hillside Group, "The hillside group website," 2012, last access: 23.06.2012. [Online]. Available: http://hillside.net

[38] P. H. Meland and J. Jensen, "Secure software design in practice," in *Proceedings of the International Conference on Availability, Reliability and Security*, Mar. 2008, pp. 1164–1171.

[39] M. Schumacher and U. Roedig, "Security engineering with patterns," in *Proceedings of the Conference on Pattern Languages of Programs*, 2001, pp. 1–17, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2001/accepted_submissions/PLoP2001/mschumacher0/PLoP2001_mschumacher0_1.pdf

[40] A. M. Braga, C. M. F. Rubira, and R. Dahab, "Tropyc: A pattern language for cryptographic software," in *Proceedings of the Conference on Pattern Languages of Programs*, 1998, pp. 1–27, last access: 23.06.2012. [Online]. Available: http://hillside.net/plop/plop98/final_submissions/P25.pdf

[41] E. B. Fernandez, J. C. Pelaez, and M. M. Larrondo-Petrie, "Security patterns for voice over ip networks," in *International Multi-Conference on Computing in the Global Information Technology*, ser. ICCGI '07. Washington, DC, USA: IEEE Computer Society, 2007, p. 33.

[42] K. Yskout, T. Heyman, R. Scandariato, and W. Joosen, "A system of security patterns," K.U.Leuven, Department of Computer Science, Report CW 469, Dec. 2006, last access: 23.06.2012. [Online]. Available: http://www.cs.kuleuven.be/publicaties/rapporten/cw/CW469.abs.html

[43] S. T. Halkidis, A. Chatzigeorgiou, and G. Stephanides, "A qualitative analysis of software security patterns," *Computers & Security*, vol. 25, no. 5, pp. 379–392, 2006.

[44] B. Elsinga and A. Hofman, "Security paradigm pattern language," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 363–380.

[45] G. Dallons, P. Massonet, J.-F. Molderez, C. Ponsard, and A. Arenas, "An analysis of the chinese wall pattern for guaranteeing confidentiality in grid-based virtual organisations," in *International Workshop on Security, Trust and Privacy in Grid Systems*. IEEE, 2007, pp. 217–222.

[46] P. Dyson and A. Longshaw, "Patterns for managing internet-technology systems," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 459–492.

[47] B. Elsinga and A. Hofman, "Control the actor-based access rights," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 233–244.

[48] A. M. Ernst, "Enterprise architecture management patterns," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–20.

[49] E. B. Fernandez, J. Ballesteros, A. C. Desouza-Doucet, and M. M. Larrondo-Petrie, "Security patterns for physical access control systems," in *Working Conference on Data and Applications Security*. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 259–274.

[50] S. Romanosky, "Security design patterns part 1," Nov. 2001, last access: 23.06.2012. [Online]. Available: http://www.cgisecurity.com/lib/securityDesignPatterns.html

[51] A. P. Moore, M. Hanley, and D. Mundie, "A pattern for increased monitoring for intellectual property theft by departing insiders," in *Proceedings of the Conference on Pattern Languages of Programs*, 2011, pp. 1–17, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2011/papers/D-6-Moore.pdf

[52] D. Riehle, W. Cunningham, J. Bergin, N. Kerth, and S. Metsker, "Password patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 279–288.

[53] S. Romanosky, A. Acquisti, J. Hong, L. F. Cranor, and B. Friedman, "Privacy patterns for online interactions," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 12:1–12:9.

[54] M. Schumacher, "Security patterns and security standards - with selected security patterns for anonymity and privacy," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 289–300.

[55] A. Cuevas, P. E. Khoury, L. Gomez, and A. Laube, "Security patterns for capturing encryption-based access control to sensor data," in *Proceedings of the International Conference on Emerging Security Information, Systems and Technologies*. Los Alamitos, CA, USA: IEEE Computer Society, 2008, pp. 62–67.

[56] S. Lehtonen and J. Pärssinen, "A pattern language for cryptographic key management," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002.

[57] S. Lehtonen and J. Pärssinen, "A pattern language for key management," in *Proceedings of the Conference on Pattern Languages of Programs*, 2001, pp. 1–13, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2001/accepted_submissions/PLoP2001/slehtonen0/PLoP2001_slehtonen0_1.pdf

[58] K. Hashizume and E. B. Fernandez, "Symmetric encryption and xml encryption patterns," in *Proceedings of the Conference on Pattern Languages of Programs*, ser. PLoP '09. New York, NY, USA: ACM, 2009, pp. 13:1–13:8.

[59] B. Blakley, C. Heath, and members of The Open Group Security Forum, *Security Design Patterns*. The Open Group, Apr. 2004, last access: 23.06.2012. [Online]. Available: www.opengroup.org/onlinepubs/9299969899/toc.pdf

[60] A. Cuevas, P. E. Khoury, L. Gomez, A. Laube, and A. Sorniotti, "A security pattern for untraceable secret handshakes," in *Proceedings of the International Conference on Emerging Security Information, Systems and Technologies*, Jun. 2009, pp. 8–14.

[61] N. Delessy and E. B. Fernandez, "Patterns for the extensible access control markup language," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–20, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2005/proceedings/PLoP2005_ndelessyandebfernandez0_1.pdf

[62] M. Schumacher, "Firewall patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 417–430.

[63] N. Delessy, E. B. Fernandez, M. M.Larrondo-Petrie, and J. Wu, "Patterns for access control in distributed systems," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2007, pp. 1–11.

[64] L. B. Jr, F. L. Brown, J. Divietri, G. D. D. Villegas, and E. B. Fernandez, "The authenticator pattern," in *Proceedings of the Conference on Pattern Languages of Programs*, 1999, pp. 1–8, last access: 23.06.2012. [Online]. Available: http://hillside.net/plop/plop99/proceedings/Fernandez4/Authenticator3.PDF

[65] E. B. Fernandez and R. Warrier, "Remote authenticator / authorizer," in *Proceedings of the Conference on Pattern Languages of Programs*, 2003, pp. 1–8, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2003/Papers/Fernandez-remote-authenticator.pdf

[66] M. Hafiz, "A collection of privacy design patterns," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 1–13.

[67] T. Okubo and H. Tanaka, "Web security patterns for analysis and design," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–13.

[68] M. Sadicoff, M. M. Larrondo-Petrie, and E. B. Fernandez, "Privacy-aware network client pattern," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–6, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2005/proceedings/PLoP2005_msadicoff0_0.pdf

[69] B. Schleinzer and N. Yoshioka, "A security pattern for data integrity in p2p systems," in *Proceedings of the Conference on Pattern Languages of Programs*, Oct. 2010, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2010/papers/schleinzer.pdf

[70] P. Sommerlad, "Reverse proxy patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, Jun. 2003, pp. 431–458.

[71] S. Romanosky, "Enterprise security patterns," 2002, last access: 23.06.2012. [Online]. Available: http://www.romanosky.net/papers/EnterpriseSecurityPatterns.pdf

[72] A. Kumar and E. Fernandez, "A security pattern for a virtual private network," in *Proceedings of the Latin American Conference on Pattern Languages of Programming*, 2010.

[73] I. A. Buckley, E. B. Fernandez, and M. M. Larrondo-Petrie, "Patterns combining reliability and security," in *Proceedings of the International Conferences on Pervasive Patterns and Applications*, IARIA Conferences. XPS (Xpert Publishing Services), 2011, pp. 144–150.

[74] C. Dougherty, K. Sayre, R. C. Seacord, D. Svoboda, and K. Togashi, "Secure design patterns," Carnegie Mellon University, Software Engineering Institute, TECHNICAL REPORT CMU/SEI 2009-TR-010, Oct. 2009, last access: 23.06.2012. [Online]. Available: www.cert.org/archive/pdf/09tr010.pdf

[75] E. B. Fernandez and J. Sinibaldi, "More patterns for operating systems access control," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, Jun. 2003, pp. 381–398.

[76] E. B. Fernandez and T. Sorgente, "A pattern language for security models," in *Proceedings of the Conference on Pattern Languages of Programs*, 2001, pp. 1–13, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2001/accepted_submissions/PLoP2001/ebfernandezandrpan0/PLoP2001_ebfernandezandrpan0_1.pdf

[77] E. B. Fernandez, "Patterns for operating systems access control," in *Proceedings of the Conference on Pattern Languages of Programs*, 2002, pp. 1–18, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2002/final/OSSecPatt7.doc

[78] E. B. Fernandez, T. Sorgente, and M. M. Larrondo-Petrie, "Even more patterns for secure operating systems," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 1–9.

[79] E. B. Fernandez and D. laRed Martinez, "Patterns for the secure and reliable execution of processes," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–16.

[80] E. B. Fernandez and G. Pernul, "Patterns for session-based access control," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 8:1–8:10.

[81] V. Gondi, "Multiple secure observers using j2ee," in *Proceedings of the Conference on Pattern Languages of Programs*, 2010, pp. 1–13, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2010/papers/gondi.pdf

[82] M. Hafiz, "Secure pre-forking - a pattern for performance and security," in *Proceedings of the Conference on Pattern Languages of Programs*, 2005, pp. 1–9, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2005/proceedings/PLoP2005_mhafiz0_2.pdf

[83] M. Hafiz, R. E. Johnson, and R. Af, "The security architecture of qmail," in *Proceedings of the Conference on Pattern Languages of Programs*, 2004, pp. 1–9, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/2004/papers/mhafiz1/PLoP2004_mhafiz1_0.pdf

[84] D. M. Kienzle and M. C. Elder, "Final technical report: Security pattern for web application development," Tech. Rep., 2002, last access: 23.06.2012. [Online]. Available: http://www.scrypt.net/~celer/securitypatterns/final%20report.pdf

[85] S. R. Kodituwakku, P. Bertok, and L. Zhao, "Aplrac: A pattern language for designing and implementing role-based access control," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2001, pp. 331–346.

[86] Q. H. Mahmoud, "Security policy: A design pattern for mobile java code," in *Proceedings of the Conference on Pattern Languages of Programs*, 2000, pp. 1–8, last access: 23.06.2012. [Online]. Available: http://hillside.net/plop/plop2k/proceedings/Mahmoud/Mahmoud.pdf

[87] H. Mouratidis, P. Giorgini, and M. Schumacher, "Security patterns for agent systems," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, Jun. 2003, pp. 399–416.

[88] P. Morrison and E. B. Fernandez, "The credentials pattern," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2006, pp. 1–4.

[89] P. Morrison and E. B. Fernandez, "Securing the broker pattern," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2006, pp. 513–530.

[90] J. L. Ortega-Arjona and E. B. Fernandez, "The secure blackboard pattern," in *Proceedings of the Conference on Pattern Languages of Programs*. New York, NY, USA: ACM, 2008, pp. 1–5.

[91] T. Saridakis, "Design patterns for fault containment," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2003, pp. 493–520.

[92] K. E. Sørensen, "Session patterns," in *Proceedings of the European Conference on Pattern Languages of Programs*. UVK - Universitaetsverlag Konstanz, 2002, pp. 301–322.

[93] M. Weiss, "Credential delegation: Towards grid security patterns," in *Proceedings of the Nordic Conference on Pattern Languages of Programs*, 2006, pp. 65–70, last access: 23.06.2012. [Online]. Available: http://hillside.net/vikingplop/vikingplop2006/VikingPLoP2006_Proceedings.pdf

[94] Y. Zhou, Q. Zhao, and M. Perry, "Policy enforcement pattern," in *Proceedings of the Conference on Pattern Languages of Programs*, 2002, pp. 1–14, last access: 23.06.2012. [Online]. Available: http://www.hillside.net/plop/plop2002/final/ZZPerry_PLOP.pdf

[95] E. B. Fernandez, S. Mujica, and F. Valenzuela, "Two security patterns: Least privilege and secure logger/auditor," in *Proceedings of the Asian Conference on Pattern Languages of Programs*, 2011, pp. 1–12, last access: 23.06.2012. [Online]. Available: http://patterns-wg.fuka.info.waseda.ac.jp/asianplop/proceedings2011/asianplop2011_submission_7.pdf

[96] O. Ajaj and E. B. Fernandez, "A pattern for the ws-trust standard for web services," in *Proceedings of the Asian Conference on Pattern Languages of Programs*, 2010, pp. 1–11, last access: 23.06.2012. [Online]. Available: http://patterns-wg.fuka.info.waseda.ac.jp/asianplop/proceedings2010/11-WS-Trust_march02-10.pdf

[97] The Internet Society, "Point-to-point tunneling protocol (pptp)," 2012, last access: 23.06.2012. [Online]. Available: http://tools.ietf.org/html/rfc2637

[98] S. T. Halkidis, N. Tsantalis, A. Chatzigeorgiou, and G. Stephanides, "Architectural risk analysis of software systems based on security patterns," *IEEE Transactions on Dependable and Secure Computing*, vol. 5, no. 3, pp. 129–142, 2008.

[99] C. Kramer and L. Prechelt, "Design recovery by automated search for structural design patterns in object-oriented software," in *Working Conference on Reverse Engineering*. Washington, DC, USA: IEEE Computer Society, 1996, p. 208.

[100] R. K. Keller, R. Schauer, S. Robitaille, and P. Pagé, "Pattern-based reverse-engineering of design components," in *International Conference on Software Engineering*. New York, NY, USA: ACM, 1999, pp. 226–235.

[101] L. Wendehals, "Improving design pattern instance recognition by dynamic analysis," in *International Conference on Software Engineering*, May 2003.

[102] R. J. Anderson, *Security Engineering: A Guide to Building Dependable Distributed Systems*, 1st ed. New York, NY, USA: John Wiley & Sons, Inc., 2001.

[103] Yahoo! Inc., "Yahoo! Design Pattern Library," last access: 23.06.2012. [Online]. Available: http://developer.yahoo.com/ypatterns/

[104] Microsoft, "Microsoft patterns & practices," last access: 23.06.2012. [Online]. Available: http://msdn.microsoft.com/en-us/practices/default

APPENDIX

The tables IV, V, VI and VII depict our classification of all software-security patterns that we collected during the literature-review process. The first dimension "Matching Aspects" is highlighted in light grey and a magnifier icon

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Access Control List (ACL) [63]	●			●	●		●			
Administrator Hierarchy [78]	●						●			
Capability [63]	●			●		●	●			
Check Point [3], [20]	●			●	●	●				
Checkpointed System [59], [83], [42]	●					●			●	
Controlled Process Creator [75], [20]	●					●	●			
Container Managed Security [22], [42]	●			●	●					
Credential [88]	●			●	●					
Encrypted Storage [18]	●					●	●			
Execution Domain [77], [20]	●					●				
Full Access With Errors [20]	●			●	●	●				
Input Validation [74]	●					●				
Multilevel Security pattern [76], [20]	●					●				
Obfuscated Transfer Object [22], [42]	●					●				
Pathname Canonicalization [74]	●					●				
Partitioned Application [18]	●						●		●	
Policy [59]	●			●	●		●			
Protected System [59]	●			●	●					
Roles [3]	●				●					
Role-Based Access Control (RBAC) [76], [20]	●				●					
Role Hierarchies [85]	●				●		●			
Sandbox [87]	●					●				
Secure Pipe [22], [42]	●					●				
Secure Communication [59], [42]	●						●			
Security Context [59]	●						●			
Secure Directory [74]	●					●				
Secure Process / Thread [78]	●			●	●	●	●			
Secure Service Facade [22], [42]	●			●			●			
Secure Session Object [22], [42]	●			●	●					
Secure Service Proxy [22]	●			●	●					
Session-Based Attribute-Based Authorization [80]	●			●	●					
Session-Based Role-Based Access Control [80]	●				●	●				
Single Session [85]	●				●	●				
Single Access Point [3], [20]	●			●						
Subject Description [59]	●			●	●			●		
Symmetric Encryption [58]	●					●	●			
XML Encryption Pattern [58]	●					●	●			
A Pattern for WS-Trust [96]		●					●	●		●
Access Controller [87]		●				●	●			
Account Lockout [18]		●		●						
Agent Authenticator [87]		●		●						
Agency Guard [87]		●				●	●			
Audit Interceptor [22]		●						●		●

Table IV
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (1).

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Authenticator [64], [59], [20]		○		●						
Authentication Enforcer [22]		○		●	●					
Authorization Pattern [76]		○			●					
Authorization Enforcer [22]		○								
Assertion Builder Pattern [22]		○		●	●					
Controlled Object Factory [20]		○			●					
Controlled Object Monitor [75] [20]		○			●	●				
Controlled Virtual Address Space [75]		○				●				
Credential Delegation [93]		○					●			
Credential Tokenizer [22]		○		●	●					
Defer to Kernel [74]		○		●	●	●				
Dynamic Service Management [22]		○								●
File Authorization [77], [20]		○			●	●	●			
Full View With Errors [3]		○				●	●		●	
Grant-Based Access Control Pattern (GBAC) [55]		○			●					
ID/Password Authentication [67]		○		●						
Information Obscurity [20]		○					●			
Intercepting Validator [22]		○				●				
Intercepting Web Agent [22]		○		●	●					
Known Partners [20]		○		●	●					
Limited Access [20]		○		●	●	●	●			
Limited View [3]		○					●			
Message Inspector [22]		○			●					
Message Interceptor Gateway [22]		○					●			
Multiple Secure Observers Using J2EE [81]		○			●		●			
Network Address Blacklist [18]		○				●	●			
Password Synchronizer Pattern [22]		○		●	●					
Policy-Based Access Control [63]		○		●	●					
Policy Delegate [22]		○					●			●
Policy Enforcement Pattern [94]		○				●	●			
Privilege Separation (PrivSep) [74]		○		●	●					
Protected Entry Points [79]		○			●					
Protection Rings [79]		○			●	●				
Secure Base Action [22]		○		●	●					
Secure Logger [22]		○						●		●
Secure Message Router [22]		○		●	●	●				
Single Sign-on Delegator Pattern [22]		○		●	●		●			
Session [3]		○				●	●			
Security Policy: A Design Pattern For Mobile Java Code [86]		○				●				
Secure Broker Pattern [89]		○		●	●					
Security Session [20]		○				●				
Session Timeout [92], [42]		○				●			●	

Table V
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (2).

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Sealed And Signed Envelope [57]		○		●		●	●			
Sealed Envelope [57]		○				●	●			
Security Association [59]		○		●		●				
Secure Builder Factory [74]		○		●		●				
Secure Chain of Responsibility [74]		○			●		●			
Secure Factory [74]		○			●		●			
Secure State Machine [74]		○				●			●	
Secure Strategy Factory [74]		○				●	●			
Secure Visitor [74]		○		●	●	●	●			
Secure Preforking [82]		○				●			●	
Virtual Address Space Access Control [77]		○				●	●			
Access Session [80]			○	●	●	●	●			
Access Control requirements [20]			○	●	●					
Actor and Role Lifecycle [47]			○	●	●		●			
Address Book [57]			○	●		●				
Administrator Objects [85]			○			●				●
Alice And Friends [57]			○	●						
Authenticated Session [18]			○	●						
Authorization [20]			○		●					
Build The Server From The Ground Up [18]			○			●			●	
Clear Sensitive Information [74]			○				●			
Client Data Storage [18]			○			●	●			
Client Input Filters [18]			○			●				
Choose The Right Stuff [18]			○			●			●	
Compartmentalization [83]			○			●			●	
Content Independent Processing [83]			○			●	●			
Controlled Execution Environment [77]			○		●	●				
Demilitarized Zone [20]			○				●			
Directed Session [18]			○			●				
Distributed Responsibility [83]			○			●	●			
Distrustful Decomposition [74]			○			●				
Document The Security Goals [18]			○							●
Document The Server Configuration [18]			○			●			●	
Enroll By Validating Out Of Band [18]			○	●		●	●		●	
Enroll Using Third-Party Validation [18]			○	●		●	●		●	
Enroll With A Pre-Existing Shared Secret [18]			○	●			●		●	
Enroll Without Validating [18]			○							●
Face-To-Face [57]			○	●						
Fault Container [91]			○			●				
Front Door [20]			○				●			
Hidden Implementation [18]			○				●		●	
Input Guard [91]			○			●				

Table VI
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (3).

Pattern Name	Matching Aspects			Security Aspects						
	Structural	Behavioral	Generic Concept	Authentication	Access Control	Integrity	Confidentiality	Non-Repudiation	Availability	Accountability
Keep Session Data In Client [92]										
Keep Session Data In Server [92]										
Key In The Pocket [57]										
Load Balancer [92], [42]										
Log For Audit [18]										
Minefield [18]										
Multilevel Secure Partitions [79]										
Output Guard [91]										
Password Authentication [18]										
Password Propagation [18]										
Patch Proactively [18]										
Privilege-Limited Role [85]										
Reference Monitor [77]										
Red Team The Design [18]										
Resource Acquisition Is Initialization (RAII) [74]										
Role Based Access [85]										
Role Validator [85]										
Secure Access Layer [3]										
Secure Assertion [18]										
Secure Channels [20]										
Server Sandbox [18]										
Session Failover [92], [42]										
Session Management [67]										
Session Scope [92]										
Seal Ring Engraver [57]										
Signed Envelope [57]										
Share Responsibility For Security [18]										
Subject Descriptor [20]										
Test On A Staging Server [18]										
The Forged Seal Ring [57]										
The Real Thing [57]										
There Is Somebody Eavesdropping [57]										
Trusted Proxy [18]										
Unique Entry of Information [83]										
Validated Transactions [18]										
Virtual Address Space Structure Selection [78]										

Table VII
SOFTWARE-SECURITY PATTERNS CLASSIFIED BY SECURITY ASPECTS AND RECOGNITION NEEDS (4).

🔍 indicates that a pattern belongs to the aspect of this dimension. The second dimension ”Security Aspects“ is highlighted with a grey background and a lock 🔒 shows which security aspects a pattern addresses. Some patterns were described by more than one publication. Therefore, we put all publications that describe the pattern with this name in the order of their publication year behind the pattern name.