

Consolidating Principles and Patterns for Human-centred Usable Security Research and Development

Luigi Lo Iacono*, Matthew Smith†, Emanuel von Zezschwitz†, Peter Leo Gorski* and Peter Nehren*

*Cologne University of Applied Sciences, Germany, {luigi.lo_iacono, peter.gorski, peter.nehren}@th-koeln.de

†University of Bonn, Germany, {smith, zezschwitz}@cs.uni-bonn.de

Abstract—We present an evaluation of usable security principles and patterns to facilitate the transfer of existing knowledge to researchers and practitioners. Based on a literature review we extracted 23 common usable security principles and 47 usable security patterns and identified their interconnection. The results indicate that current research tends to focus on only a subset of important principles. The fact that some principles are not yet addressed by any design patterns suggests that further work on refining these patterns is needed. We developed an online repository, which stores the harmonized principles and patterns. The tool enables users to search for relevant patterns and explore them in an interactive and programmatic manner. We argue that both the insights presented in this paper and the repository will be highly valuable for students for getting a good overview, practitioners for implementing usable security and researchers for identifying areas of future research.

I. INTRODUCTION

Three seminal papers are seen as the origin of the research domain of Usable Security and Privacy. Zurko and Simon's: "User-Centered Security" [1], Adams and Sasse's: "Users Are Not the Enemy" [2] and Whitten and Tygar's: "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0" [3]. All argued that users should not be seen as the problem to be dealt with, but that security experts need to communicate more with users, and adopt user-centered design approaches. Many studies have shown that it is worth making this effort because design faults often lead to security issues or frustration among users (e.g., [2], [3], [4], [5], [6]). From this, usable security principles and patterns have been suggested to guide developers in building usable and secure software systems. More recently, researchers have argued that developers also need to be the focus of usable security research [7], [8], [9], [10], [11] since they are only human too and need as much, if not more, help and guidance than end users, since any mistake they make is amplified. However, the body of knowledge built up over two decades of usable security research has not yet been systematized in a way to give developers easy access to the principles and design patterns needed to create usable and secure software.

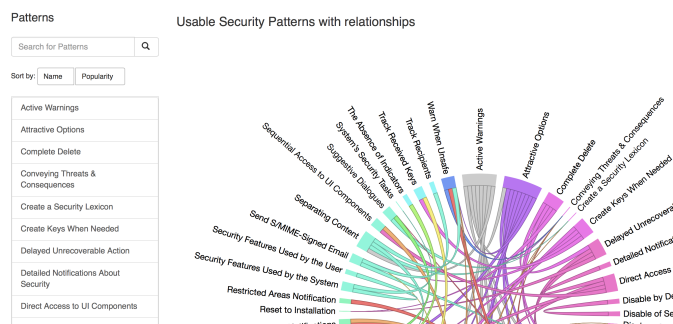


Fig. 1. Our tool facilitates the search of relevant principles and patterns in various ways. One example is an interactive dependency graph that is built upon interconnections derived from an evaluation of the present patterns.

Principles are general rules on the highest level of abstraction, which should be followed by a system's architecture. In contrast, patterns relate to concrete implementation problems and provide actionable solutions on a lower level of abstraction. Such a guidance is a crucial prerequisite in order to support software developers with the information sources required for implementing effective security [7]. Our research on usable security principles and patterns reveals, however, that they are scattered throughout the usable security literature and are very inhomogeneous. This makes it difficult for non-specialized or inexperienced developers to grasp and access that knowledge. This also presents a problem for researchers since, as we will show, certain areas of research are overrepresented while others have received little attention to date.

This paper provides (1) *the first systematic overview and evaluation* of usable security principles and usable security patterns and proposes (2) *a standardized way* of representing such knowledge. We identify (3) *links and dependencies* within and between the principles and patterns which offers insights into which patterns can help with which principles and what principles are covered by which patterns. Finally, we offer the community (4) *an online repository* (see Figure 1) that stores the principles and patterns and allows users to explore them based on our evaluation as well a full-text search, a dependency graph, a tag-based categorization and programmable interfaces. We hope this repository will be further extended by the community and become a useful tool for students to get an overview of usable security principles and patterns, for researchers to position their work and identify areas where further work is

needed and for practitioners to find the relevant design patterns to help them implement the usable and secure principles they wish to fulfill.

II. METHODOLOGY

Since the first notable publications in the usable security domain in 1996 [1], many principles and patterns have been developed and proposed as guidance for developing usable security mechanisms. However, there exists neither a comprehensive collection of this state of the art nor a homogeneous description. We argue that this lack prevents developers from accessing this knowledge easily and researchers from analyzing the present state in order to derive new insights. By conducting an exhaustive analysis of all proposed principles and patterns we extract recurring attributes used in the descriptions, focusing on the evaluation of the contained knowledge. The precise methodology is shown in Figure 2.

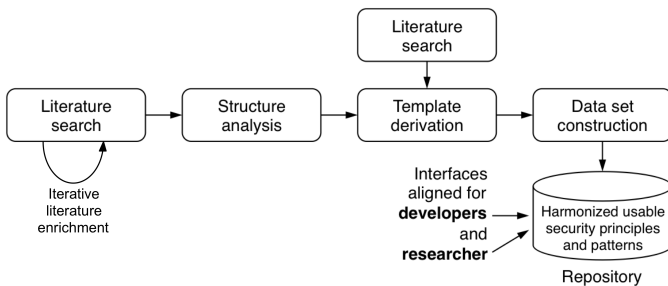


Fig. 2. Methodology used in the present evaluation.

In an initial step, we conducted a broad literature search with the goal to identify as many publications containing usable security principles and patterns as possible. The literature was selected both by going through the complete proceedings of usable security venues SOUPS, CHI, USEC, HAISA and EuroUSEC and the top security conferences S&P, EuroS&P, CCS, USENIX Security and NDSS, as well as using search engines. Search terms used herein were "usable security" in combination with "principle" and "pattern" as well as frequently used synonyms like "heuristic" or "guideline". We then proceeded to iteratively add to the literature search based on keywords, if given by a literature source, and references, until we felt confident that we have reached a good coverage of published work. We found a good coverage of principles as the 23 identified candidates are described by four publications (cf. Table II) in a period of five years during 2002 to 2007. This rather small amount of principles is comprehensible due to their abstract nature. Moreover, in the two more senior disciplines, usability and security, the amount of known principles is comparably small. In contrast, the more concrete proven solutions provided by patterns are available for various problems in distinct settings making them numerous. By applying the same methodology we were able to identify 47 patterns. As we assume more patterns to be existing, we developed an open and extensible repository for the community to contribute (see Sections V and VII for further details and discussions).

To gather unique artifacts we conducted a content-related structure analyses discussed in subsection III-A for principles and in subsection IV-A for patterns. Since the field of usable research for developers is still very young we concentrated on principles and patterns for end-users.

On the basis of the collected literature, we correlated the descriptions of the principles and the patterns respectively. The goal was to identify recurring attributes that could serve as a common ground to document both artifacts in a formalized and harmonized manner. For the principles we created our own formalization since we found none in the literature. For the patterns, we adapt formalization from software engineering. Further details about how we derived the templates are described in subsection III-B and subsection IV-B. We then developed and stored the systematized data set of usable security principles and usable security patterns into an interactive web-based repository: <https://das.th-koeln.de/usecured>.

III. USABLE SECURITY PRINCIPLES

Principles are the most abstract tool focusing on the system's architecture level. Principles compile rough concepts and do not offer concrete implementations or solutions to problems. Hence, they can be adopted in the very early design phases of the software development process. In the light of usable security, related principles can guide the design and contribute to an implementation of improved security mechanisms that are tailored to the needs of the targeted user group.

Principles are an accepted tool in both individual domains. Usability principles are better known under the term *Usability Heuristics*. These heuristics are commonly used by experts to evaluate the usability of a target object. Additionally, developers make use of them to guide their designs contributing to an implementation without usability flaws. The most commonly used usability principles are the heuristics provided by Nielsen [12]. One of Nielsen's heuristics is e.g. *Consistency and standards*. Considering this principle means constructing all system's elements in a consistent manner so that a user can be sure that all words, situations, or actions regarding a particular element mean the same thing. The adherence to this principle can be observed in almost all modern operating systems. The elements in the user interface have a consistent vocabulary, structure and appearance.

Security principles for the information technology domain have first been introduced by Saltzer and Schroeder in 1975 [13]. They analyzed what proven security strategies known from the physical domain can also be adapted to computer systems. One of their identified principles is e.g. *Fail-safe Defaults*. It recommends basing access decisions on permission rather than exclusion. Considering this principle means that in case of an error the default is lack of permission, which is safer default than lack of exclusion. The definition of firewall filtering rules is guided by this principle. In 2002 an additional set of ten principles have been proposed by Viega and McGraw [14], while they confirmed a whole bunch of the Saltzer and Schroeder principles by repeating them. Still, Vega and McGraw contributed some unique ones including e.g. the *Promote privacy* principle that advices to minimize the data gathered, processed, stored and transmitted by the system. The Viega and McGraw principles emphasize that the Saltzer and Schroeder principles are still relevant today, although a contemporary analysis also raises some thoughts on the topicality of some of the principles as well as the completeness of the present set [15].

Usable security principles are in the intersection of usability principles and security principles. As such, usable security

principles should neither repeat a usability principle nor a security principle by simply setting the focus of the context to the other discipline. Such an approach would only bloat the set of usable security principles making it hard to sort through. Thus, we defined that usable security principles must be unique to both domains. This definition will serve as criterion to decide whether a principle obtained from the literature analysis can be classified as usable security principle.

A. Literature Analysis

Following the methodology introduced above, an extensive literature review has been conducted in order to collect and analyze the available usable security principles. The obtained results are discussed in chronological order of the publication's appearance date.

Whitten and Tygar define four conditions that should be met by usable security systems [3]. The recommendations have been extended by Chiasson et al. [16]. Overall, six recommendations can be summarized to principles as follows [3], [16]:

Completion. Users should be able to tell the system when their tasks are completed.

Error Prevention. Help users to avoid making dangerous errors.

Feedback. The system should give informative feedback of the current security status.

Satisfaction. Make interfaces as comfortable as possible to support users satisfaction.

Support. Help users to successfully perform security tasks.

Transparency. Make users aware of the security tasks they need to perform.

Each principle is denoted by a unique declarative name followed by a short description of its intent. This is the common scheme of laying out principles as has, e.g., been adopted for the usability principles by Nielsen [12] and for the security principles by Saltzer and Schroeder [13].

However, we argue that the six principles are based on common usability heuristics. For example, *Error Prevention* and *Satisfaction* have been adopted one by one from [12]. Thus, the principles from Whitten et al. [3] and Chiasson et al. [16] must not be assigned to the usable security principles set.

Garfinkel defines more specific principles for aligning usability and security in systems which are based on research from security practitioners in industry and academia [17].

Consistent Controls and Placement. Security-related controls in graphical user interfaces should be standardized, so that similar functionality is presented in a similar manner and in a consistent location.

Consistent Meaningful Vocabulary. Security information must be standardized, used consistently and understandable for users.

Good Security Now. Ensure that systems offering some security features are deployed now, rather than leaving these systems sitting on the shelf while researchers try to develop "perfect" security systems for deployment later.

Least Surprise. Ensure that the system acts in accordance with the user's expectations. Computers should not surprise users when these expect the computer to behave in a secure manner.

No External Burden. Minimize impact of security systems on system-external users. Otherwise, users could be adversely affected and could be forced to stop using that security system.

Provide Standardized Security Policies. Provide a few standardized security configurations that can be audited, documented, and taught to users. Avoid security policies, options and choices, which overwhelm users. But policies and configurations for experts should be available optionally.

Ka-Ping Yee contributes design principles for secure interaction design [18], [19] that can be generally adopted when designing user interfaces for interacting with security mechanisms. In Yee's principles actors are users or programs and authorities are entities with the abilities of taking particular actions [20]:

Appropriate Boundaries. The interface should expose, and the system should enforce, distinctions between objects and between actions along boundaries that matter to the user.

Clarity. The effect of any security-relevant action must be clearly apparent to the user before the action is taken.

Expected Ability. The interface must not generate the impression that it is possible to do something that cannot actually be done.

Explicit Authorization. A user's authorities must only be provided to other actors as a result of an explicit action that is understood by the user to imply granting.

Expressiveness. The interface should provide enough expressive power to describe a safe security policy without undue difficulty; and to allow users to express security policies in terms that fit their goals.

Identifiability. The interface should enforce that distinct objects and distinct actions have unspoofably identifiable and distinguishable representations.

Path of Least Resistance. To the greatest extent possible, the natural way to do any task should also be the secure way. Grant e.g. the least of authority while finding the most usable workflow to do tasks.

Revocability. The interface should allow the user to easily revoke authorities that the user has granted wherever revocation is possible. Users should be able to revoke such consent and therefore reduce authorities to access their resources if possible.

Self-awareness. Maintain accurate awareness of the user's own authority to access resources. Users should be made aware of the risks of their own authority caused by their access rights.

Trusted Path. The interface must provide an unspoofable and faithful communication channel between the user and any entity trusted to manipulate authorities on the user's behalf. A user's communication channels to other entities have to be protected, especially if the entity is trusted to access resources or to manipulate authorities.

Visibility. Interfaces should visualize active authorities and actors to give users the option to check and reconfigure their system.

The need of every principle has been validated with a real-life example. Moreover, Yee concludes that if one of his principles is violated, security vulnerabilities would occur.

In a further publication of Yee that deals with the alignment of usability and security two more principles have been outlined [20]. Computers do not know exactly what the user expects and considers acceptable when other entities behave in their name. **Security by Admonition** means that in some cases users should be asked whether another entity is allowed to perform an action. That can lead to problems when such confirmations are presented too often. Therefore, Yee suggests that **Security by Designation** should be used whenever possible. This principle means that an entity starts with a minimal set of abilities. The abilities were gradually expanded by user actions, which lead to an extension of authority of this entity over time.

During a survey with over 300 participants that aimed at understanding security features in operation systems, Furnell et al. have collected various criteria that needs to be considered when including security features in end-user software [21]:

Convenience. Although visibility is important, the provision of security should not become so prominent that it is considered inconvenient or intrusive. Warnings, e.g., should not be shown too often. Otherwise, it is possible that users ignore them or disable the respective security features.

Locatability. Users need to be able to find the features they need. Security mechanisms have to be easy to find. If it takes too long to find them, users might give up and remain unprotected.

Understandability. Options and descriptions should be presented in a manner that is meaningful to the intended user population. It is necessary that users are able to understand options and descriptions of the security system. If possible, help and support should be offered for beginners.

Visibility. The system should give a clear indication of whether security is being applied. The security status of the system should be clearly but not intrusively visible for users. This reminds users to activate security features.

To be in conformance with the other principles, we changed the original names as introduced by Furnell et al. from verbs to nouns. Again, these principles match with others in the general usability research field. The *Visibility* principle, e.g., matches with one of Nielsen's ten heuristics for usability [12], [22], [23], [24]. The *Understandability* principle matches with Garfinkel's principles *Consistent Meaningful Vocabulary* and *Consistent Controls and Placement*.

The collection by Chiasson et al. [16] are related to persuasive technology [25], which focuses on how to motivate users to perform actions in a desired way. The authors outline four principles [16]:

Conditioning. Using positive reinforcement to encourage the desired behavior.

Expertise. Incorporating signs of expertise such as experience, knowledge and competence to gain credibility with the users.

Reciprocity. Harnessing the human tendency to return favors.

Reduction. Making the desired path one of least resistance.

These principles are, e.g., implemented in many antivirus applications [16]. Users get notifications in case malware has been detected in order to emphasize the benefit of the

tool. Likewise, such software programs promote credibility by informing on the security status or signature database updates.

The following principles are based on the previous four principles and ensure that administrators are supported when there is need for making important decisions [16]:

Administrators should

- reliably and promptly be made aware of the security tasks they must perform.
- be able to figure out how to successfully perform those tasks.
- be able to tell when their task has been completed.
- have sufficient feedback to accurately determine the current state of the system and the consequences of their actions.
- be able to revert to a previous system state if a security decision has unintended consequences.
- be able to form an accurate and meaningful mental model of the system they are protecting.
- be able to easily examine the system from different levels of encapsulation in order to gain an overall perspective and be able to effectively diagnose specific problems.
- be able to easily seek advice and take advantage of community knowledge to make security decisions.

The interface should

- facilitate interpretation and diagnosis of potential security threats.
- encourage administrators to address critical issues in a timely fashion.

This collection focuses on interfaces for security professionals. That is, potentially entire networks could be left vulnerable to attacks if these principles are violated. Still, the level of abstraction of these principles is not considered abstract enough to warrant the inclusion to the principles' repository. Instead, they should be collected in a more specific guideline, which includes how administrators should be informed properly.

The results obtained from the literature analysis in the field of usable security principles emphasize that many efforts have been undertaken to document research results in terms of general rules and advice for supporting developers as well as researchers. Moreover, it shows that many researchers are influenced by the same literature, which causes the creation of similar principles. The ten principles of [16] are tailored to one particular user group, system administrators. They are an adaption of other more general principles present in the gathered collection. The six principles [16], [3] presented in the very beginning are not usable security principles in a strict sense. Instead, they are usability principles that have been paraphrased to focus on security mechanisms. The pairs of *Visible* and *Visibility* as well as *Reduction* and *Path Of Least Resistance* have been identified as duplicates. Some principles like the ones from [16] are already documented in a structured manner including also references. Still, there does not exist a common template for documenting the various principles in a harmonized view.

B. Derived Template

To derive a common template for usable security principles, the collected principles have been analyzed from two

distinct perspectives. First, the different structures used in the literature have been extracted and distilled—attribute-wise—to the greatest common denominator. Second, the principle descriptions have been filtered for the contained attributes. Again, the obtained sets have been reduced down to the ones contained in the intersection of all sets. Finally, the derived template has been correlated with the set of attributes common to all principle descriptions. This evaluation leads us to the following set of attributes with which usable security principles can be described and managed in a harmonized manner (see Table I).

TABLE I. THE DERIVED TEMPLATE FOR DOCUMENTING USABLE SECURITY PRINCIPLES IN A SYSTEMATIZED FORM.

Name	Unique name for the principle
Sources	Sources and references of the described principle
Synonyms	Known synonyms or names in other languages for the principle
Intent	Description of the principles' intentions
Motivation	Description of the context or circumstances that motivate to apply the principle
Examples	Known uses and illustrations of the described principle
Tags	Keywords providing further describing and categorizing information
Log History	Field for storing logging events, such as the latest updates of the principle

The first and most important attribute is the name, since this is the identifying component. Then, the sources are listed in order to preserve where the principle originates. Furthermore, the template includes a field in which synonyms can be stored. In case alias names do exist, each synonym must include references to the source, in which it is introduced. The following three attributes intent, motivation and examples describe the principle. The last two attributes are required for operative purposes and ease of management.

C. Extracted Principles

After de-duplication and relevance filtering, 23 from the 44 gathered principles remain as unique candidate principles. Table II shows all 23 distilled principles and enriches them with attributes systematizing the observations made during the analysis. It contains a column, which emphasizes what patterns respect a certain principle that will be explained later. Still, another column examines the relation to general usability and security principles. Most of the usable security principles show to have a tendency towards one of the both disciplines only and in some cases they even show to be a simple instantiation of a general usability principle to a security-focused application context. We claim that the latter candidates should not be explicitly listed as usable security principle as this unnecessary bloat would render the tool complex and unhandy.

This approach is arguable, though, and thus the repository, which is introduced in section V, contains all of the 23 principles for the time being. With this paper and the provided repository we want to stimulate a broader discussion in the community. This will ensure that various viewpoints from distinct users spanning from practitioners to researchers can be assembled and taken into account. By means of a comments

thread for each principle, a discussion within the community should guide the process of decision-making on whether none, some or all of the questionable principles will be marked as deprecated and finally discard from the repository.

The interactive online repository offers additional information that for the sake of brevity is not contained in Table II. We will address these aspects in the overall discussion.

IV. USABLE SECURITY PATTERNS

Patterns provide more concrete and actionable solutions to common and recurring problems than the abstract advice and recommendations offered by principles. Patterns will, in fact, in many cases adhere to one or more principles as baseline for their proven solutions. The original pattern approach was developed and introduced by the architect (of buildings) Christopher Alexander in order to document proven architectural designs in a standardized structure [26]. Nowadays patterns are used in various fields, including software, usability and security engineering. There are different types of patterns, which are used in distinct stages of the system development process [27]. This paper focuses on design patterns, which are used during the software design phase.

In the human computer interaction (HCI) community, proven user interface (UI) solutions have been transformed into usability patterns—often also referred to as interaction design patterns—, which have been collected in numerous pattern catalogs and have been published via diverse channels. An overview of the various catalogs is given in [28]. In the presence of long-running tasks the *Progress Display* [29] pattern, e.g., advice to inform users about the progress and help them estimate the time remaining. The adoption of this pattern can be observed in almost any dialog providing feedback on the download of a (large) file. Furthermore, it gets apparent that this patterns respects the *Visibility of system status* principle in its proposed solution.

Patterns have been evolved in the information security domain likewise. A security pattern describes a proven solution to a recurring problem of controlling—i.e. preventing, detecting or correcting—a set of specific threats by means of security controls in a given context [30]. Since the first notable publication by Yoder and Barcalow in 1997 [31], many other patterns appeared and there now exists a pattern language that categorizes and unifies the variety of security patterns [32]. The *DoS Safety* [33] pattern, e.g., addresses the threat of Denial of Service (DoS) attacks on the system architecture level. It advises to protect against DoS attacks by setting resource limits. One example is the implementation of a multi-threaded server. In order to avoid DoS vulnerabilities by following the *DoS Safety* pattern one should limit the total number of threads by means of a thread pool.

As security patterns, usable security patterns focus on providing solutions to problems of controlling a set of specific threats while paying explicit attention to provide usable solutions.

A. Literature Analysis

As there are many more usable security patterns available from the literature than principles, this section will not mention

TABLE II. OVERVIEW OF THE CANDIDATE USABLE SECURITY PRINCIPLES IDENTIFIED BY THE LITERATURE REVIEW. TO ASSESS EACH CANDIDATE’S ELIGIBILITY, THEIR RELATIONSHIP WITH USABILITY AND SECURITY PRINCIPLES AS WELL AS THEIR ADOPTION BY USABLE SECURITY PATTERNS HAVE BEEN ANALYZED. NOT ALL PRINCIPLES ARE CLEARLY DISTINGUISHABLE FROM EXISTING PRINCIPLES AND RESPECTED BY AT LEAST ONE PATTERN, INDICATING A RESEARCH DEMAND IN RESPECT TO THE RELEVANCE AND COMPLETENESS OF THE CURRENT STATE.

Usable Security Principle Candidate	Tendency towards Usability/Security	Respected by Pattern
Appropriate Boundaries [19]	●/○	—
Clarity [19]	●/○	Active Warnings, Attractive Options, Conveying Threats & Consequences, General Notifications About Security, Separating Content, Warn When Unsafe
Conditioning [16]	●/○ Strong link to user experience	Immediate Options
Consistent Controls and Placement [17]	●/○ (cf. <i>Consistent Meaningful Vocabulary</i> and <i>Understandability</i> as well as <i>Consistency and standards</i> [12])	Direct Access to UI Components, Distinguish Security Levels, Immediate Options, The Absence of Indicators, Sequential Access to UI Components
Consistent Meaningful Vocabulary [17]	●/○ (cf. <i>Understandability</i> as well as <i>Consistency and standards</i> [12] and <i>Match between system and the real world</i> [12])	Create a Security Lexicon, Disclose Significant Deviations, General Notifications About Security, Informative Dialogues, Security Features Used by the System, Security Features Used by the User, System’s Security Tasks
Convenience [21]	●/○ (cf. <i>Expressiveness</i> as well as <i>Aesthetic and minimalist design</i> [12])	Active Warnings, Email-Based Identification and Authentication, Migrate and Backup Keys
Expected Ability [19]	●/○	—
Expertise [16]	●/○ (cf. <i>Identifiability</i>)	Security Features Used by the System, Security Features Used by the User, System’s Security Tasks
Explicit Authorization [19]	●/○ Strong link to privacy	—
Expressiveness [19]	●/○ (cf. <i>Convenience</i> as well as <i>Aesthetic and minimalist design</i> [12])	Quick Description of UI Components
Good Security Now [17]	○/●	Create Keys When Needed, Key Continuity Management
Identifiability [19]	●/○ (cf. <i>Trusted Path</i> and <i>Visibility</i>)	Attractive Options, Distinguish Security Levels, Send S/MIME-Signed Email, Separating Content, Track Received Keys
Least Surprise [17]	●/○	Complete Delete, Disable by Default, Reset to Installation
Locatability [21]	●/○ (cf. <i>Consistent Controls and Placement</i> and <i>Consistent Meaningful Vocabulary</i> as well as (cf. <i>Consistency and standards</i> [12])	Direct Access to UI Components, Explicit Item Delete, Indirect Access to UI Components, Localization of Specific Areas
No External Burden [17]	●/●	Track Recipients
Path of Least Resistance [19]	○/● (cf. <i>Least privileges</i> [13])	Attractive Options, Install Before Execute, Providing Recommendations, Reset to Installation, Suggestive Dialogues
Provide Standardized Security Policies [17]	●/● (cf. <i>Consistent Controls and Placement</i> , <i>Consistent Meaningful Vocabulary</i> and <i>Understandability</i> as well as <i>Flexibility and efficiency of use</i> [12])	Warn When Unsafe
Reciprocity [16]	●/○	Warn When Unsafe
Revocability [19]	○/● Strong link to privacy	Delayed Unrecoverable Action
Self-awareness [19]	●/● (cf. <i>Visibility</i> as well as <i>Visibility of system status</i> [12])	—
Trusted Path [19]	○/● (cf. <i>Trusted Path</i> security control contained in opensecurityarchitecture.org)	Separating Content, Distinguish Between Run and Open
Understandability [21]	●/● (cf. <i>Consistent Controls and Placement</i> and <i>Consistent Meaningful Vocabulary</i> as well as <i>Flexibility and efficiency of use</i> [12])	Failing Safely, General Notifications About Security, Immediate Notifications, Levels of Severity, Redundant Notifications
Visibility [19]	●/○ (cf. <i>Visibility of system status</i> [12])	Active Warnings, Detailed Notifications About Security, Distinguish Internal Senders, Explicit User Audit, Immediate Notifications, Localization of Specific Areas, Noticeable Contextual Indicators, Redundant Notifications

them all individually. Instead, they will be grouped and discussed from the viewpoint of the overarching category. An overview of all gathered patterns is shown in Figure 3.

Garfinkel structures his collection of usable security design patterns in three categories [17]:

- User Visibility and Sanitization Patterns
- Identification and Key Management Patterns
- Patterns for Promoting Overall Secure Operation

The patterns contained in the *User Visibility and Sanitization Patterns* category are related to issues concerning the transparency and control of security-relevant actions. One deals, e.g., with the deletion of sensitive data. If data is to be deleted from systems such as operating systems or web browsers then usually the visual representation of it is deleted only. The actual content is not erased from the storage. It becomes just invisible for users. Consequently, they might assume that sensitive data has been deleted but actually it has not. This often matters when storage devices get disposed, even as part of larger systems. Another main topic in this class deals with is visibility of user-generated information. Users should have the opportunity to inspect all their personal information carried out by the system. Furthermore, information should be deletable from where it is shown.

In the *Identification and Key Management Patterns* category approaches for mail signature and encryption and for secure messaging in general are contained. While the approach of certificates and certification authorities is well known in organizations and for web sites operators, it fails with end-users. Exceedingly few private users apply email signature and encryption with digital certificates to their messages.

In the *Patterns for Promoting Overall Secure Operation* category Garfinkel collects concepts for many different operations, which combine usability and security aspects. Some of them are based on the work of Ka-Ping Yee [19]. One of them advocates that organizations should provide a lexicon with definitions of used security-related terms. This pattern adheres to the Consistent Meaningful Vocabulary principle and enriches it by more concrete instructions on how to implement such a vocabulary. Other included patterns provide solutions on how to inform users when systems or objects behave in an unexpected manner. In this context, warnings should, e.g., not be hidden automatically so that users have to close them actively. Still, other patterns focus on viruses and malware. One pattern calls for an execution of programs only after installation in order to reduce the loopholes into a system.

Some of the patterns described by Garfinkel deal with security warnings. As security status indicators are very vital parts of every security system, their usability is particularly relevant. Egelman's patterns focus on security warnings only [34]. With his patterns he fills a gap in Garfinkel's collection that does not contain any pattern on the design of security warnings. All patterns are described extensively and in detail. Moreover, Egelman lists trade-offs of patterns and outlines how attackers could potentially exploit them.

The collection of Arteaga et al. offers more general solutions in terms of user interface patterns for designing information security feedback [35], [36], [37]. They focus on the representation of information security feedback, which is

presented to users from any kind of system. They are based on the HCI-S criteria from [38]. HCI-S stands for Human-Computer Interaction for Security and adapts traditional HCI concepts to improve usability of security interfaces. The patterns are categorized in three different topics, which are based on basic parts of user interfaces. The first class is called *Informative Feedback* and collects patterns for presenting useful information to users. Examples for such useful information are available security features and how to use them, the detection of threats and the general security status of the system. The second class is denoted as *Interaction Feedback* in which patterns for establishing navigation and operation of feedback are collected. They treat problems like how to activate or deactivate security features. The last class, which is called *Interactive Feedback*, gathers patterns for specifying the security feedback of systems. The patterns included therein describe how to design auditiv and visual notifications.

B. Derived Template

The template for the usable security patterns has been derived following the same methodology as for deriving the usable security principles template. Additionally, the available literature on pattern templates was considered. Blakley and Heath define a minimal set of attributes for describing a pattern [39]. According to this, a pattern should contain a name as identifying feature, a description of the problem, an evaluated solution to solve the problem and it should outline the possible consequences. Based on this, the following set of attributes has been derived, by which the gathered 47 usable security patterns can be documented and managed in a harmonized manner (see Table III).

In respect to the first three fields, the introduced pattern template is equal to the principle template. The patterns related fields Context, Problem, Solution, Examples, Implementation and Consequences are the core ones, that every pattern needs to provide. The fields Dependencies and Relationships store link information and are used for constructing a pattern language as will be described subsequently. Further relationships with other artifacts are maintained by the template structure as well. The template contains references to principles, guidelines, checklists and use cases. This new view on the patterns highlights what principles have been used in which patterns or which patterns can be used to implement a particular principle. The reverse view is also available in the principles Table II. These views should make it easier for researchers to identify, which areas of research have already progressed from the principle level of abstraction to the concrete pattern level of abstraction as well as helping developers find relevant patterns to achieve usable security principles. We believe that this part of the evaluation should prove valuable both for driving further research as well as transferring research results into practice. As for the principle template, the last two entries are required for technical maintenance purposes in respect to the online repository.

V. ONLINE REPOSITORY

On the basis of the introduced principle and pattern templates as well as the pattern language, an online repository has been developed and deployed, in order to provide easy access to the presented knowledge. The repository contains the

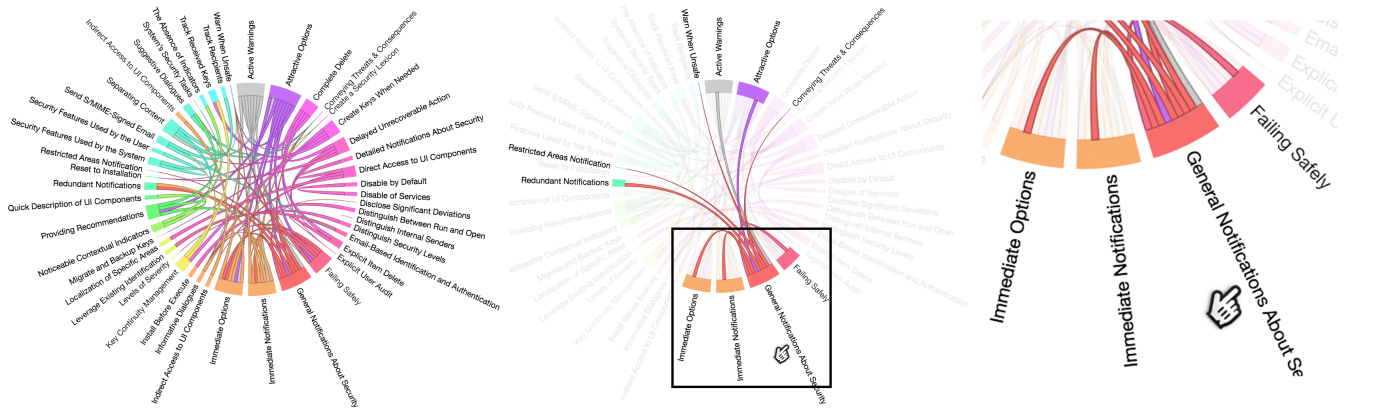


Fig. 3. The interactive visualization shows all 47 patterns and their relationships amongst each other (left). The visualization emphasizes related patterns when selecting one specific pattern by hovering over its unique name with the mouse pointer (center, right).

TABLE III. THE DERIVED TEMPLATE FOR DOCUMENTING USABLE SECURITY PATTERNS IN A SYSTEMATIZED FORM.

Name	Unique name of the pattern
Sources	References to the sources of the pattern
Synonyms	Known synonyms or alias names
Context	Description of the situation in which a specific problem occurs, and in which the pattern is applicable
Problem	Description of the problem in the given context, including the influencing factors and requirements (eg. security or usability objectives, conditions, restrictions) arising thereby
Solution	Description of a proven design, which enables to meet the given requirements
Examples	Known uses and illustrations of the pattern
Implementation	Detailed information about the pattern including, e.g., the specification of functional / non-functional requirements or references to architectural concepts or implementations
Consequences	Advantages and disadvantages of the pattern, caused by factors or requirements that conflict with each other, and that should be considered before using the pattern
Dependencies	Dependencies on other patterns
Relationships	Links to patterns that address a similar problem or which may be used in combination with the pattern
Principles	Links to principles that represent major objectives, which are to be achieved with the pattern
Guidelines	Links to guidelines in which the pattern can be implemented
Check lists	Links to checklists for verifying if the pattern has been implemented correctly
Use cases	Links to use cases in which the pattern should be considered
Tags	Keywords providing further describing and categorizing information
Log History	Field for storing logging events, such as the latest updates of the principle

23 principles and the 47 patterns and can be accessed at the following address: <https://das.th-koeln.de/usecured>

We developed and implemented different access modalities to meet the diverse requirements and preconditions of the distinct targeted user groups. For students and practitioners search and filter functions can be used to find relevant patterns in the first place. By means of various interactive visualizations the navigation and exploration of the repository is further enhanced. This enables the discovery of related principles and patterns that might possibly be of relevance. Researchers and developers are equipped with a programmatic user interface in terms of an application programming interface in conjunction with a specified data structure representing the templates in JSON format. The API allows performing data analytics on the content of the repository as well as supports the integration of the artifacts into third party software such as usable security assessment tools.

The platform finally allows registered users to comment on the principles and patters as well as suggest additions to create a living repository of usable security knowledge. Figure 4 shows Egelman’s pattern *Active Warning* which was transferred to the developed pattern template.

VI. DISCUSSION

The gathered usable security principles and patterns offer a comprehensive and standardized data set, forming a core tool for researchers, students and practitioners. Beyond making the principles and patterns easily accessible, the present evaluation enables us to perform various examinations of the current state of the art. For instance, we inspected which patterns adopts which principles, performed an analysis on relationships present amongst the distinct patterns and proposed a classification in order to group the patterns. These are three examples showing the potential benefits stemming from the performed collection, consolidation and evaluation of usable security principles and patterns.

Patterns

Search for Patterns

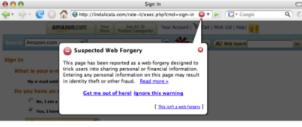
Sort by: Name Popularity

- Active Warnings
- Attractive Options
- Complete Delete
- Conveying Threats & Consequences
- Create a Security Lexicon
- Create Keys When Needed
- Delayed Unrecoverable Action
- Detailed Notifications About Security
- Direct Access to UI Components
- Disable by Default
- Disable of Services
- Disclose Significant Deviations
- Distinguish Between Run and Open
- Distinguish Internal Senders
- Distinguish Security Levels
- Email-Based Identification and Authentication
- Explicit Item Delete
- Explicit User Audit
- Falling Safely
- General Notifications About Security
- Immediate Notifications
- Immediate Options
- Indirect Access to UI Components
- Indirect Access to UI Components

Patterns are based on scientific sources.

Active Warnings

[en], [de]

Name	Active Warnings
Source	(Egelman 2009)
Synonyms	None
Context	Passive warning styles that do not interrupt the user may go unnoticed and thus be rendered useless. Likewise, a warning may be passive if it can be dismissed without the user taking notice of it.
Problem	Some warnings fail in very critical situations because they were not prominent enough for the user to notice them.
Solution	Active Warnings should be used to grab users' attention by interrupting their primary tasks, thus forcing them to acknowledge the warnings by taking an action in order to proceed.
Examples	
Implementation	Active warnings must be designed to interrupt the primary task by either replacing the content users were expecting with the warning message, or by drawing attention away from the expected content.
Consequences	By interrupting users' primary tasks and forcing them to make a decision, significantly more users paid attention to the warnings and were ultimately protected from the phishing attack.
Dependencies	None
Relationships	[Attractive Options] [Immediate Notifications] [Conveying Threats & Consequences] [General Notifications About Security] [Immediate Options] [Separating Content]
Principles	[Convenience] [Clarity]
Guidelines	None
Check lists	None
Use cases	None
Tags	Active Warnings, Immediate Notifications, Warnings
Log history	[12/21/2015]: Added to repository
References	Egelman, Serge. 2009. <i>Trust Me: Design Patterns for Constructing Trustworthy Trust Indicators</i> . ProQuest. http://reports-archive.adm.cs.cmu.edu/anon/isr2009/CMU-ISR-09-110.pdf .

Comments

There are no comments yet.

Type a comment here...

Fig. 4. Web-based user interface of the patterns repository showing the developed template and some of the means for searching and crawling the contained patterns.

A. Derived Pattern Language

Some of the attributes contained in the introduced usable security pattern template build-up a pattern language. The *Relationship* attribute stores the unique and identifying name of usable security patterns contained in the repository, to which a relationship exists.

The relationships have been analyzed for all collected and documented patterns (see Figure 3, left). This allows more advanced search functionality to be offered. The search results include the most relevant patterns and with the contained relationships, a user can quickly verify whether the related patterns do also or even fit better to her needs, or she obtains further insights on how to improve her search.

As shown in Figure 3 (right), advanced (interactive) visualizations can be developed based on the pattern language as well. The depicted dependency wheel contains all patterns and interconnects them according to their relationships. Using this, alternative patterns which might be relevant for the pursued development or research activity can be spotted more easily by following the emphasized relationships. In order to get a better understanding on how the interactivity of the dependency wheel works, Figure 3 (center) shows the appearance of the wheel, when a pattern is hovered by the mouse pointer. In case of Figure 3 (center) the mouse points to the *General*

Notifications About Security pattern and highlighting the nine related patterns.

As noted above, this enhanced view on the systematized knowledge may provide new insights. Patterns that only have a small number of relations with other patterns or even no relation at all need further attention. Those indications might lead to results that further enhance the current set of patterns.

B. Classification of Usable Security Patterns

As a further means for analyzing the completeness of the compiled usable security patterns set, we tried to apply a classification to the available patterns. As already mentioned, Garfinkel introduced a classification to group the patterns documented in his dissertation [17]. It contains three classes in which the patterns are grouped as *Identification and Key Management Patterns*, *User Visibility and Sanitization Patterns* or *Patterns for Promoting Overall Secure Operation*. We kept this classification in the first place and analyzed whether the other patterns will add to these classes. Only the term *Visibility* seemed not accurate enough. The contained patterns deal with transparency-enhancing patterns for end-users. Thus, we slightly renamed the category to *User Transparency and Sanitization Patterns*.

Egelman has not provided a categorization for his proposed patterns [34]. This is due to the patterns being focused on one domain only, namely security warnings. Thus, the Egelman patterns do not fit in any of the classes defined by Garfinkel. Thus we added an according category design and the implementation of *Security Warnings*.

Muñoz-Arteaga et al. [35] clustered their proposed patterns into three groups. The names of the categories are related to their task in a graphical user interface. Since most of the patterns deal with notifications and feedback, we grouped them together with the patterns on security warnings.

C. Principle Coverage by Patterns

One question arising when examining the available usable security principles and patterns is how well used, relevant and interconnected they are, in order to identify areas where more research is needed. In order to answer this question, we inspected each individual pattern and derived whether the solution it provides adopts one or more of the principles. We documented such relationships as *Principles* in the patterns template (see Table III) by including according references to the patterns. Table II shows all collected principles. It contains a column, which emphasizes what patterns respect a certain principle.

The goal of this analysis has been to identify principles that do not have any relation to patterns and vice versa. As can be seen from Table II this is true for the principles *Appropriate Boundaries*, *Expected Ability*, *Explicit Authorization* and *Self-awareness*. This view provides insights about the completeness and relevance of the available principles and patterns. One reason for the missing adoptions of principles might be the lack of appropriate patterns. In this case, research activities need to focus on the current set of usable security patterns in order to extend it as required. Another cause might be the lack of relevance of the principle. It might even be the case that a

principle without any adoption by a pattern should be struck from the list at some point. Again, this would require further research to analyze if the principles that have not been adopted need to be excluded from the current data set.

VII. LIMITATIONS

There are inherent limitations in our research approach, which need to be acknowledged. First of all, the insights presented in this paper are mainly based on a literature review. Therefore, the results rely on the availability and accessibility of previously published research. In addition, the presented results may be influenced by a selection bias. To minimize such general limitations, we applied several countermeasures. For example, we prevented subjective selection of research papers by applying a systematic research approach with predefined sources, keywords and inclusion criteria.

At the same time, this systematic approach may have reduced the number of included papers. Even though we are confident that we considered the most relevant conferences and search terms, it is likely that we missed out relevant publications. Likewise, some of the presented principles or patterns may be invalid in specific context and some recommendations may be outdated, already. As a consequence, we emphasize that the presented set is neither complete nor final and should rather be seen as the basis for a novel systematization approach. We hope that the community will accept this approach and that our online repository will be used to add, revise and delete principles and patterns in the future.

Finally, the proposed distinction into principles and patterns might have limitations, too. Future research will show if more categories are required. For example, *guidelines* may make up a third category of recommendations which are too specific for general principles but not specific enough to serve as patterns.

VIII. CONCLUSION AND FUTURE WORK

This paper systematized both usable security principles and patterns and the hitherto unexplored connections between these two. Based on a literature analysis we derived a set of templates and an evaluation linking them in order to obtain a global view of the available knowledge in a standardized description. With this we offer a view on which principle have been adopted by which patterns and which pattern considers which principles. Our evaluation shows that although there are a wide variety of usable security principles and patterns, the coverage of relationships between principles and patterns is quite diverse. There even exist principles such as *Explicit Authority* or *Self-awareness* that are not addressed by any pattern yet and patterns which do not fit to the given principles such as *Leverage Existing Identification* or *Restricted Areas Notification*. We also implemented our evaluation as an interactive online repository to allow students to get an overview of usable security principles and patterns, researchers to present their work in a standardized manner and identify new areas of research and practitioners to search for relevant principles and patterns.

Finally, we do not argue that the presented set of principles and patterns is complete. We assume that further relevant data can be found in publications, which do not focus on methodological aspects in the first place. For example, still

unidentified usable security principles could be found as part of general recommendation and implication sections. Therefore, we hope that this paper will start a discussion within the community and that the online repository will serve as a useful tool to revise and extend the set of usable security patterns and usable security principles in the long run.

ACKNOWLEDGMENT

This work has been partially funded by the German Federal Ministry for Economic Affairs and Energy (Grant no. 01MU14002), the German Federal Ministry of Education and Research within the funding program "Forschung an Fachhochschulen" (contract no. 13FH016IX6), the Ministry of Culture and Science of the German State of North Rhine-Westphalia within the funding program "Digitale Sicherheit" and by the ERC Grant 678341: Frontiers of Usable Security.

REFERENCES

- [1] M. E. Zurko and R. T. Simon, "User-Centered Security," in *NSPW '96: Proceedings of the 1996 workshop on New security paradigms*, ser. NSPW '96. ACM, Sep. 1996, pp. 27–33. [Online]. Available: <https://doi.org/10.1145/304851.304859>
- [2] A. Adams and M. A. Sasse, "Users Are Not the Enemy," *Communications of the ACM*, vol. 42, no. 12, pp. 40–46, Dec. 1999. [Online]. Available: <https://doi.org/10.1145/322796.322806>
- [3] A. Whitten and J. D. Tygar, "Why Johnny Can't Encrypt: A Usability Evaluation of PGP 5.0," in *Proceedings of the 8th Conference on USENIX Security Symposium*, ser. SSYM'99. USENIX Association, 1999, pp. 14–14. [Online]. Available: <https://www.usenix.org/conference/8th-usenix-security-symposium/why-johnny-cant-encrypt-usability-evaluation-ppg-50>
- [4] A. J. DeWitt and J. Kuljis, "Aligning Usability and Security: A Usability Study of Polaris," in *Proceedings of the Second Symposium on Usable Privacy and Security*, ser. SOUPS '06. New York, NY, USA: ACM, 2006, pp. 1–7. [Online]. Available: <https://doi.org/10.1145/1143120.1143122>
- [5] I. Kirlappos and M. A. Sasse, "What usable security really means: Trusting and engaging users," in *International Conference on Human Aspects of Information Security, Privacy, and Trust*, ser. HAS 2014. Cham: Springer International Publishing, jun 2014, pp. 69–78. [Online]. Available: https://doi.org/10.1007/978-3-319-07620-1_7
- [6] M. A. Sasse, S. Brostoff, and D. Weirich, "Transforming the "Weakest Link" - a Human/Computer Interaction Approach to Usable and Effective Security," *BT Technology Journal*, vol. 19, no. 3, pp. 122–131, Jul. 2001. [Online]. Available: <https://doi.org/10.1023/A:1011902718709>
- [7] Y. Acar, M. Backes, S. Fahl, D. Kim, M. L. Mazurek, and C. Stransky, "You get where you're looking for: The impact of information sources on code security," in *2016 IEEE Symposium on Security and Privacy (SP)*. IEEE, May 2016, pp. 289–305. [Online]. Available: <https://doi.org/10.1109/SP.2016.25>
- [8] Y. Acar, S. Fahl, and M. L. Mazurek, "You are not your developer, either: A research agenda for usable security and privacy research beyond end users," in *IEEE Cybersecurity Development (SecDev)*. IEEE, nov 2016, pp. 3–8. [Online]. Available: <https://doi.org/10.1109/SecDev.2016.013>
- [9] S. Fahl, M. Harbach, H. Perl, M. Koetter, and M. Smith, "Rethinking SSL development in an appified world," in *Proceedings of the 2013 ACM SIGSAC Conference on Computer & Communications Security*, ser. CCS '13. New York, NY, USA: ACM Press, 2013, pp. 49–60. [Online]. Available: <https://doi.org/10.1145/2508859.2516655>
- [10] P. L. Gorski and L. Lo Iacono, "Towards the usability evaluation of security apis," in *Proceedings of the 10th International Symposium on Human Aspects of Information Security & Assurance*, ser. HAISA, jul 2016, pp. 252–265. [Online]. Available: <http://www.cscan.org/openaccess/?paperid=287>
- [11] M. Green and M. Smith, "Developers are not the enemy!: The need for usable security apis," *IEEE Security & Privacy*, vol. 14, no. 5, pp. 40–46, Sept 2016. [Online]. Available: <https://doi.org/10.1109/MSP.2016.111>

- [12] J. Nielsen and R. Molich, "Heuristic Evaluation of User Interfaces," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '90. New York, NY, USA: ACM, 1990, pp. 249–256. [Online]. Available: <https://doi.org/10.1145/97243.97281>
- [13] J. H. Saltzer and M. D. Schroeder, "The protection of information in computer systems," *Proceedings of the IEEE*, vol. 63, no. 9, pp. 1278–1308, Sept 1975. [Online]. Available: <https://doi.org/10.1109/PROC.1975.9939>
- [14] J. Viega and G. McGraw, *Building Secure Software - How to Avoid Security Problems the Right Way*, 1st ed., ser. Professional Computing Series. Addison-Wesley, 2001.
- [15] R. E. Smith, "A contemporary look at saltzer and schroeder's 1975 design principles," *IEEE Security & Privacy*, vol. 10, no. 6, pp. 20–25, Nov 2012. [Online]. Available: <https://doi.org/10.1109/MSP.2012.85>
- [16] S. Chiasson, R. Biddle, and A. Somayaji, "Even experts deserve usable security: Design guidelines for security management systems," in *Workshop on Usable IT Security Management (USM'07)*, 2007. [Online]. Available: https://cups.cs.cmu.edu/soups/2007/workshop/Design_Guidelines.pdf
- [17] S. L. Garfinkel, "Design Principles and Patterns for Computer Systems That Are Simultaneously Secure and Usable," Ph.D. dissertation, Massachusetts Institute of Technology, may 2005. [Online]. Available: <http://simson.net/thesis/>
- [18] B. D. Payne and W. K. Edwards, "A Brief Introduction to Usable Security," *IEEE Internet Computing*, vol. 12, no. 3, pp. 13–21, May 2008. [Online]. Available: <https://doi.org/10.1109/MIC.2008.50>
- [19] K.-P. Yee, "User interaction design for secure systems," in *Proceedings of the 4th International Conference on Information and Communications Security*, ser. ICICS '02. Springer Berlin Heidelberg, 2002, pp. 278–290. [Online]. Available: https://doi.org/10.1007/3-540-36159-6_24
- [20] —, "Aligning security and usability," *IEEE Security & Privacy*, vol. 2, no. 5, pp. 48–55, Sept 2004. [Online]. Available: <https://doi.org/10.1109/MSP.2004.64>
- [21] S. M. Furnell, A. Jusoh, and D. Katsabas, "The challenges of understanding and using security: A survey of end-users," *Computers & Security*, vol. 25, no. 1, pp. 27 – 35, Feb. 2006. [Online]. Available: <https://doi.org/10.1016/j.cose.2005.12.004>
- [22] J. Nielsen, "Enhancing the Explanatory Power of Usability Heuristics," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, ser. CHI '94. ACM, 1994, pp. 152–158. [Online]. Available: <https://doi.org/10.1145/191666.191729>
- [23] —, "Usability inspection methods," in *Conference companion on Human factors in computing systems*, ser. CHI '94. ACM, 1994, pp. 413–414. [Online]. Available: <https://doi.org/10.1145/259963.260531>
- [24] R. Molich and J. Nielsen, "Improving a Human-computer Dialogue," *Communications of the ACM*, vol. 33, no. 3, pp. 338–348, Mar. 1990. [Online]. Available: <https://doi.org/10.1145/77481.77486>
- [25] B. J. Fogg, "Persuasive technology: Using computers to change what we think and do," *Ubiquity*, vol. 2002, no. December, Dec. 2002. [Online]. Available: <https://doi.org/10.1145/764008.763957>
- [26] C. Alexander, S. Ishikawa, and M. Silverstein, *A Pattern Language - Towns, Buildings, Construction*. Oxford University Press, 1977.
- [27] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*, 1st ed., ser. Professional Computing. Addison-Wesley, Nov. 1994.
- [28] A. Dearden and J. Finlay, "Pattern languages in hci: A critical review," *Human-Computer Interaction*, vol. 21, no. 1, pp. 49–102, jan 2006. [Online]. Available: https://doi.org/10.1207/s15327051hci2101_3
- [29] H. Röder, "Specifying usability features with patterns and templates," in *2012 First International Workshop on Usability and Accessibility Focused Requirements Engineering*, ser. UsARE, June 2012, pp. 6–11. [Online]. Available: <https://doi.org/10.1109/UsARE.2012.6226790>
- [30] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann, and P. Sommerlad, *Security Patterns: Integrating Security and Systems Engineering*. John Wiley & Sons, Dec. 2005.
- [31] J. Yoder and J. Barcalow, "Architectural patterns for enabling application security," in *The 4th Pattern Languages of Programming Conference*, ser. PLoP '97, 1997. [Online]. Available: <http://hillside.net/plop/plop97/Proceedings/yoder.pdf>
- [32] E. Fernandez-Buglioni, *Security Patterns in Practice: Designing Secure Architectures Using Software Patterns*, ser. Wiley Software Patterns Series. John Wiley & Sons, Apr. 2013.
- [33] M. Hafiz and R. E. Johnson, "Evolution of the mta architecture: The impact of security," *Software: Practice and Experience*, vol. 38, no. 15, pp. 1569–1599, Dec. 2008. [Online]. Available: <https://doi.org/10.1002/spe.v38:15>
- [34] S. Egelman, "Trust Me: Design Patterns for Constructing Trustworthy Trust Indicators," Ph.D. dissertation, Carnegie Mellon University, School of Computer Science Carnegie Mellon University Pittsburgh, PA 15213, apr 2009. [Online]. Available: <http://reports-archive.adm.cs.cmu.edu/anon/isr2009/CMU-ISR-09-110.pdf>
- [35] J. Muñoz-Arteaga, R. M. González, and J. Vanderdonckt, "A classification of security feedback design patterns for interactive web applications," in *2008 The Third International Conference on Internet Monitoring and Protection*, ser. ICIMP '08, June 2008, pp. 166–171. [Online]. Available: <https://doi.org/10.1109/ICIMP.2008.21>
- [36] J. Muñoz-Arteaga, R. M. González, M. V. Martín, J. Vanderdonckt, F. Á. Rodríguez, and J. M. González-Calleros, "A method to design information security feedback using patterns and HCI-security criteria," in *Proceedings of the Seventh International Conference on Computer-Aided Design of User Interfaces*, ser. CADUI 2008. Springer London, jun 2008, pp. 283–294. [Online]. Available: https://doi.org/10.1007/978-1-84882-206-1_26
- [37] J. Muñoz-Arteaga-Arteaga, R. M. González, M. V. Martín, J. Vanderdonckt, and F. Álvarez-Rodríguez, "A methodology for designing information security feedback based on User Interface Patterns," *Advances in Engineering Software*, vol. 40, no. 12, pp. 1231 – 1241, 2009. [Online]. Available: <https://doi.org/10.1016/j.advengsoft.2009.01.024>
- [38] J. Johnston, J. H. P. Eloff, and L. Labuschagne, "Security and Human Computer Interfaces," *Computers & Security*, vol. 22, no. 8, pp. 675–684, Dec. 2003. [Online]. Available: [https://doi.org/10.1016/S0167-4048\(03\)00006-3](https://doi.org/10.1016/S0167-4048(03)00006-3)
- [39] B. Blakley and C. Heath, *Security Design Patterns*, ser. Technical Guide. The Open Group, apr 2004, vol. G031. [Online]. Available: <http://pubs.opengroup.org/onlinepubs/9299969899/toc.pdf>