

A knowledge-based approach to security requirements for e-health applications

S. Dritsas¹, L. Gymnopoulos², M. Karyda², T. Balopoulos²,
S. Kokolakis², C. Lambrinouidakis² and S. Katsikas²

¹*Department of Informatics,
Athens University of Economics and Business, GR-10434, Greece
sdritsas@aueb.gr*

²*Laboratory of Information and Communication Systems Security (Info-Sec-Lab),
Department of Information and Communication Systems Engineering,
University of the Aegean, Samos, GR-83200, Greece
{lazaros.gymnopoulos, mka, tbalopoulos, sak, clam, ska}@aegean.gr*

Abstract

This paper introduces a knowledge-based approach for the security analysis and design of e-health applications. Following this approach, knowledge acquired through the process of developing secure e-health applications is represented in the form of security patterns; thus, it is made available to future developers. In this paper we present a set of security patterns that was developed based on the aforementioned approach. Security requirements for this set of patterns have been identified following a security and privacy analysis. The security patterns have been designed on the basis of a security ontology that was developed for this purpose. The ontology allows all concepts of importance and their relationships to be identified. The paper also describes the validation of the developed ontology, and compares the approach employed to other relevant methods in the domain of secure application development.

Keywords

Security requirements, application development, ontology, security patterns, e-health applications

1. Introduction

Security analysis and design involves the identification of security requirements and the design of solutions that address these requirements in an efficient and effective manner. This is a knowledge-intensive task, typically performed by security experts. In most cases, however, solutions provided for a specific application are not documented in a form that

would allow their application in other similar applications; thus, expert knowledge acquired through security analysis and design remains tacit.

Nevertheless, in application domains where security is a critical factor, such as the e-health domain, applying ad-hoc solutions may have hazardous effects on the trustworthiness of the applications. Consequently, there is a need for a systematic method of recording and re-using security solutions in critical application domains.

This paper addresses the issue of identifying and fulfilling security requirements for critical applications, as those in the e-health domain. The approach followed herein takes a knowledge management stance, using ontologies as a vehicle for managing different security requirements and their corresponding solutions. Specifically, an ontology is developed and used as the basis for building security patterns that can be applied by e-health application developers.

The goal of this paper is to show how a long-lasting difficulty in application development can be addressed through the use of a corresponding ontology. Moreover, the approach described in this paper allows expert knowledge to be exploited by application developers who otherwise would not have access to it. To achieve this, the paper proposes a set of security patterns; each pattern encapsulating a specific security problem, or requirement, a well-established way to address this problem and other information concerning the application context. All this information allows developers to make informed decisions in the process of developing secure applications. The patterns included in this paper pertain to applications in the electronic health (e-health) domain. However, it must be noted that a generic structure for describing the patterns has been adopted, that allows their use in other domains as well.

The paper is structured as follows: section two presents the security analysis for e-health applications. Based on this analysis, a set of common security and privacy requirements is identified. Section three describes the mapping of a subset of these requirements to security solutions, in the form of a set of security patterns. Section four describes the ontology that has been developed in order to design the structure of the security patterns. Section five provides an overview of related work, discussing it in relation to the approach employed in this paper. The last section provides the overall conclusions and suggestions for further research.

2. Identifying security and privacy requirements in the context of e-health

Electronic healthcare services, or e-health services, aim to improve our quality of life. A typical e-health system, as the one considered for the purposes of this paper, provides medical staff (doctors and nurses in a hospital or a healthcare establishment) with the ability to remotely access patients' health data, usually through a network that can be a

public one, such as the Internet. Depending on the severity of their health condition, patients may not need to be hospitalized, but they may be able to have a normal life while their medical condition is being monitored and sometimes controlled by healthcare professionals.

In this context, *security*, meaning the protection of data integrity, availability, authenticity and confidentiality and *privacy*, meaning compliance with personal data protection regulations, are critical factors towards achieving users' trust and acceptance of e-health systems, given the highly sensitive nature of personal health data. In the following paragraphs, a basic set of security and privacy requirements is identified; these requirements should be fulfilled in order to develop, deploy and use a reliable, secure and trustworthy e-health system.

2.1 Threat Analysis

In the context of any e-health system or application it is important to ensure that the personal data are kept secure and available to the entities who are authorized to access it (Gritzalis et al. 1999). This is often not easy to achieve since many data related to a patient are delivered or processed through public networks, particularly through the Internet. In the following analysis we take into consideration threats that originate from deliberate actions and aim to violate the fundamental security attributes, namely confidentiality, integrity and availability as well as privacy. Following a threat analysis of an e-health system, we have identified the basic privacy and security requirements that should be considered in the process of developing a secure e-health application.

2.1.1 Threats to data confidentiality

Malicious users gaining access to personal data constitute a major threat to data confidentiality in e-health systems, since the main type of data that can be maliciously accessed is patients' personal and health data; this threat applies to data that are transmitted over public networks (i.e. Internet), as well as when data are stored in the servers of an e-health system.

Hence, the major components of an e-health system that might be subject to confidentiality threats are the communication channel between the patients' terminal equipment (e.g. sensors) and the hospital and/or the medical staff equipment, the communication channel between e-health application servers and the end-users' (patients') application and, finally, the data stored in the e-health server or data stored in the end-users' equipment (e.g. a patient's personal computer).

2.1.2 Threats to data integrity

The realization of threats against the integrity of data transmitted by or stored in the e-health system by malicious users results in unauthorized altering of the data. Attacks that can realize such threats interfere with the transmissions, so that the legitimate recipient receives data different from those sent by the originator, or modify data stored in the system. The type of data that is subject to these threats as well as the components of an e-health system that might be subject to integrity threats are the same as in the threats to confidentiality case.

2.1.3 Threats to data authenticity

By violating data authenticity in e-health applications a malicious user may counterfeit false medical data and deceive recipients into believing that the data come from a different originator (which the recipient takes as the authentic originator). These kinds of threats apply both to entity identity (e.g. identity spoofing) and to data origin (e.g. IP address spoofing). This threat involves forging the part of the data where the originator is identified (usually in the headers). Repudiation is an alternative of this type of threat that consists in refusing authorship (ownership) or the contents of data previously sent. The impact of such threats, which may result in falsified medical data, can be as severe as leading to life-threatening situations.

2.1.4 Threats to availability

By compromising availability a malicious user may influence the e-health system (or a specific component) in a manner that impedes the offering of expected services to the end-users. The most common type of this threat is Denial of Service (DoS). In the cases where information provided through the e-health application is needed on a real-time basis, such threats, if realized, can have very severe consequences with regard to the health condition of many people.

2.1.5 Threats to Privacy

There are several characteristics of medical data that underline the high importance of privacy in an e-health system context (EC 2003). The most obvious is the type of data that the devices used in such environments store and process. The collection of data from these devices amplifies the tracking and profiling capabilities of personal data collectors and, therefore, poses a serious threat against personal data. Furthermore, a user in an e-health environment may not be fully aware of what, when, and which devices are monitoring her, what kind of data they are gathering, where these data are stored, to whom they are transmitted, and by whom they are accessed and processed.

Threats to privacy in the context of an e-health system can be associated with: a) cases where e-health system's components (e.g. sensors) act transparently to users, b) enhanced storage capabilities provide easier access, processing and transmission of personal and health data, c) advances in data mining techniques can increase the amount and types of personal data that are captured and analyzed, d) data communication in an e-health environment may disclose personal data.

Based on the threats identified for e-health systems or applications, as described previously, the following paragraphs describe a basic set of security and privacy requirements that must be fulfilled.

2.2 Identifying Security Requirements

E-health applications with web interface which use public networks for data communication are faced with a multitude of threats, as described above. In the first place, the application must be able to identify the user by using some form of authentication. Furthermore, it is essential that the authentication process is secure and that the session handling mechanism used to track authenticated users is equally well protected. Designing secure authentication and session management mechanisms are just a couple of the issues facing e-health application designers and developers. Additionally, preventing parameter manipulation and the disclosure of sensitive data are other top issues. Some of the most important security requirements for the development of an e-health application are depicted in Figure 1:

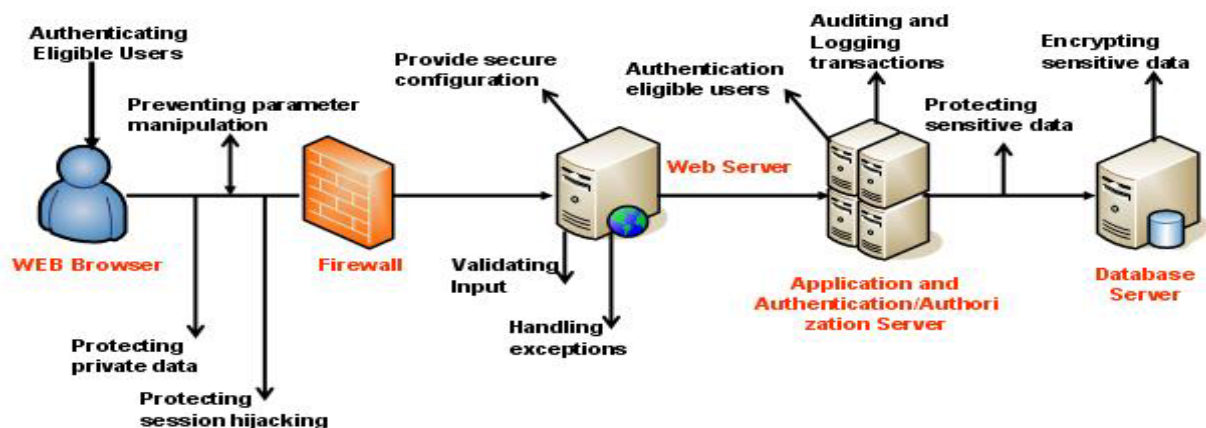


Figure 1: Basic security requirements related to an e-health application

Table 1 summarizes a basic set of security requirements with regard to web-based e-health systems. The set has been built using the threat analysis that was presented previously and the relevant literature (EC 2003; Gritzalis et al. 1999).

Security Requirement	Short description
<i>Authentication</i>	E-health applications need to incorporate mechanisms based on which users can access specific functions in accordance with their credentials.
<i>Authorization</i>	E-health applications should control access to resources and operations. Particularly with regard to medical data processing, sophisticated access control rules should be applied.
<i>Confidentiality</i>	E-health applications need to employ all necessary cryptographic services to provide confidentiality of patients' personal data. Furthermore, data confidentiality should also be provisioned for medical data transmitted over public networks.
<i>Availability</i>	Access to e-health applications and Quality of Service (QoS) should be retained at a maximum level.
<i>Integrity</i>	Specific mechanisms should be in place so that the integrity of the application itself (e.g. source code) as well as of the data processed and stored by the e-health application (particularly with regard to patients' personal and sensitive data) is maintained.
<i>Configuration Management</i>	Configuration management pertains to how the application handles security-related operational issues.
<i>Input Validation</i>	Input validation pertains to the way the e-health application filters, scrubs, or rejects input before additional processing.
<i>Parameter Management Handling</i>	Form fields, query string arguments, and cookie values are frequently used as parameters for the application. Parameter manipulation pertains to both how the application protects tampering of these values and to how it processes input parameters.
<i>Accountability</i>	Appropriate mechanisms should be in place so that adequate segregation of duties is enforced
<i>Auditing and Logging</i>	The e-health application should provide all the appropriate means in order to be able to record all security-related events.

Table 1: Security requirements for e-health applications

2.3 Identifying Privacy Requirements

Privacy is an abstract concept and its perception by individuals is determined by four independent factors: a) the information recipient, b) the intended use of the information, c) the information sensitivity and d) the context in which the disclosure occurs. Currently many countries provision privacy protection at a constitutional level. In countries where

laws, regulations and guidelines have been adopted to protect and ensure privacy, it is important to ensure that the processing of personal data should be only allowed only if it is necessary for carrying out authorized tasks, whenever explicit consent of the data subject exists, or on occasions when this is required by the law.

The preservation of privacy in an e-health application context is of paramount importance, due to the nature of the data processed by such applications. The set of basic privacy requirements listed below has been identified through the analysis of fair information practices with regard to privacy protection (OJEC 1995):

- Notice: End-users of the e-health application should always be aware that personal data are being collected. Furthermore, they should know the type of that data and the purpose for which they are collected.
- Choice and consent: The end-users should be left free to choose whether the collection of their personal data should take place or not.
- Anonymity and Pseudonymity: Specific techniques should be placed in order to protect the identity of users when this is not required and when the users do not consent.
- Access and Resource: End-users should have access to their data. Additionally, a regulatory framework should be adopted in order to protect the users against parties that do not comply with the current principles or laws.

3. Addressing the security and privacy requirements with security patterns

The concept of patterns has its roots in architecture. Alexander used this concept while writing on urban planning (Alexander et al. 1977). Later on, in 1994, the use of design patterns emerged as an approach to capturing and reusing software design expertise. Gamma et al. (1995) defined design patterns as “descriptions of communicating objects and classes that are customized to solve a general design problem in a particular context”. Under this perspective, software patterns depict the past knowledge refined from experienced developers about a particular aspect of each domain of study. In conjunction with the evolution of software patterns, the concept of security patterns was introduced in order to incorporate security techniques and best practices (e.g. specific countermeasures) during the software development process.

3.1 Security Patterns

Patterns use existing, well-proven experience in software development and help to promote effective software design practices. Each pattern deals with a specific, recurring problem in the design of software and can be used to build applications with specific properties. The use of security patterns increases the security know-how and awareness and assists software designers in incorporating security mechanisms and techniques into the software development process, by using known solutions and practices in recurring security problems. Hence, security patterns

assist developers in adopting effective security solutions (e.g. specific security countermeasures that have been tested) and in using them in the intended way.

Different types of structures for designing a pattern have been proposed. To design the structure of the security patterns that is proposed herein we follow the approach of (Schumacher et al., 2006). We have built a security ontology to derive important security attributes and the relations among them, as described in the next section. In this way, the pattern structure comprises the following attributes, that have been derived from the developed ontology: a) *name of the pattern*, b) *overview* of the security problem the pattern addresses, c) general description of a well-established *solution* that meets the identified problem, d) specific *security requirements* it addresses, e) *assets* protected when the specific solution is applied, f) *threats* that are addressed, g) *vulnerabilities* that are encountered and h) *related security patterns* which, in conjunction with the main pattern, can offer a more robust security mechanism for addressing the same security problem.

3.2 Security Patterns for the e-health domain

The following Tables present the security patterns based on the structure presented above. It should be noted that the developed patterns do not cover all possible security and privacy requirements that may be encountered during the process of designing and developing an e-health application. They constitute, however, a representative set of possible requirements that illustrates how security patterns can be used by application developers.

<i>Pattern Name</i>	<i>Authentication</i>
<i>Overview</i>	The authentication pattern allows users of e-health applications, including patients and medical staff to access the services of the e-health application without having to re-authenticate themselves continuously.
<i>Problem</i>	Each user should have access to each component of the e-health application according to his/her role and specific privileges.
<i>Solution</i>	Apply specific access control mechanisms.
<i>Requirements</i>	Confidentiality, Privacy, Accountability, Access Control
<i>Asset</i>	E-health Application Components, User Identities, User Credentials
<i>Threats</i>	Identity spoofing, Unauthorized access, Access to files, OS access
<i>Vulnerabilities</i>	Data File Exposure, Source Code Exposure, Elevation of Privilege, File Manipulation, File Missing Authorization, Application Missing Authorization, Weak Authorization.
<i>Related Patterns</i>	Single Sign-On, Authentication Related Patterns

Table 2: The Authentication Security Pattern

<i>Pattern Name</i>	<i>Credentials Propagation</i>
<i>Overview</i>	The Credentials Propagation pattern provides for increased security by requiring that e-health's user authentication credentials be verified before access is granted to that user's data.
<i>Problem</i>	The application server should authenticate the user and restrict access only to the data associated with the authenticated user.
<i>Solution</i>	The e-health application should be split into a front end and a back end. The front end is responsible for interacting with the user while the back end performs transaction processing and manages individual user account data. The front end has no direct access to user account data. When the front end initiates a transaction or accesses user account data, it presents the credentials to the back end. The back end validates the user's credentials before allowing any access to the user's data or performing any transactions on behalf of the user.
<i>Requirements</i>	Integrity, Confidentiality, Accountability (indirectly), Privacy
<i>Asset</i>	User credentials-data, Application, OS, User Identities
<i>Threats</i>	Intrusion, Identity Spoofing, Man-in-the-middle attacks
<i>Vulnerabilities</i>	Weak authentication schemes, insecure credentials propagation
<i>Related Patterns</i>	Authentication related patterns

Table 3: The Credentials Propagation Security Pattern

<i>Pattern Name</i>	<i>Secure Storage</i>
<i>Overview</i>	The Secure Storage pattern suggests the use of encryption to allow sensitive or otherwise security-critical data (health data of a patient) to be securely stored on the server hosting the e-health application.
<i>Problem</i>	The main reason for securing sensitive data on the e-health application host is because the latter cannot be trusted.
<i>Solution</i>	Use encryption techniques to protect data that is stored on the machine on which the e-health application runs. Using encryption ensures that sensitive data will not be revealed. Using message authentication codes or hash functions ensures that the data cannot be tampered with on the client.
<i>Requirements</i>	Integrity, Privacy, Confidentiality
<i>Asset</i>	Source code of the application, data manipulated by the application, user identities, personal and medical data.
<i>Threats</i>	Intrusion, attacks at the application level, misuse of the applications.
<i>Vulnerabilities</i>	Sensitive data stored in plain text, inadequate protection of user accounts, bugs in the application.
<i>Related Patterns</i>	Encrypted Communication, Secure Remote Authentication

Table 4: The Secure Storage Security Pattern

<i>Pattern Name</i>	<i>Secure Communication</i>
<i>Overview</i>	The Secure Communication pattern suggests the use of encryption to allow sensitive or otherwise security-critical data to be securely transmitted over an open network.
<i>Problem</i>	Secure medical data exchange over public networks.
<i>Solution</i>	This pattern requires the use of encryption schemes in order to protect the data transmitted over an insecure network.
<i>Requirements</i>	Confidentiality, Integrity, Accountability (Non-repudiation), Privacy
<i>Asset</i>	Data transmitted over the network, user credentials in case of unsecured remote authentication.
<i>Threats</i>	Communications interception, threat to data confidentiality and integrity.
<i>Vulnerabilities</i>	Unencrypted data transmission, remote authentication with unencrypted user credentials
<i>Related Patterns</i>	Data storage, Secure Remote Authentication

Table 5: The Secure Communication Security Pattern

<i>Pattern Name</i>	<i>Session Management (protection of specific session)</i>
<i>Overview</i>	The Session Management pattern ensures that users will not be able to skip around within a series of session regarding a specific function of the e-health application.
<i>Problem</i>	Many applications often have to collect a great deal of data from a user in order to complete a single transaction-session. This approach can be vulnerable to attacks. An attacker can use URL spoofing techniques to jump between different tasks, in an attempt to bypass specific tasks.
<i>Solution</i>	The Session Management pattern exposes a single task to the end user. Information about the active session is maintained by the e-health application. Session information refers to the current user task; thus, as the user navigates through the system, the currently selected task is maintained in the session data.
<i>Requirements</i>	Confidentiality, Accountability, Integrity, Privacy
<i>Asset</i>	Application components, application itself, OS, user data, user identities
<i>Threats</i>	Session hijacking, session replay, any type of spoofing
<i>Vulnerabilities</i>	Applications bugs, insecure session procedures, weak authentication
<i>Related Patterns</i>	Authentication related patterns, Logging patterns, Encrypted communication

Table 6: The Session Management Security Pattern

<i>Pattern Name</i>	<i>Hidden Implementation</i>
---------------------	------------------------------

<i>Overview</i>	The Hidden Implementation pattern limits an attacker's ability to discern the internal workings of the e-health application.
<i>Problem</i>	Current applications tend to provide the attacker with a great deal of information about both the standard components and the custom-made elements. As a result, it can be very easy for a potential attacker to assess whether an attack is likely to be successful or not.
<i>Solution</i>	The Hidden Implementation pattern forces examination of all components of the e-health application from an attacker point of view in order to find clues that might provide information about the internal workings of the application itself.
<i>Requirements</i>	Integrity, Availability, Privacy
<i>Asset</i>	Application source code, application data
<i>Threats</i>	Intrusion, bugs in the application, misuse of the application
<i>Vulnerabilities</i>	Bugs of the application.
<i>Related Patterns</i>	No related patterns have been identified

Table 7: The Hidden Implementation Security Pattern

<i>Pattern Name</i>	<i>Anonymization</i>
<i>Overview</i>	The Anonymization pattern provides the mechanisms that enforce the hiding of the identities of the e-health application's users. This pattern is very critical for the context of the e-health application domain.
<i>Problem</i>	Many applications require the real identities of the participated users in order to satisfy the authorization and accountability requirements. This situation may not be desirable in the health context and especially when a user, e.g. a patient wants to keep her real identity hidden.
<i>Solution</i>	In the context of the e-health application specific mechanisms should be adopted in order to protect the real identities of users and their personal data. These mechanisms should be based on well-known anonymity or pseudonymity techniques and should not, at the same time compromise the authorization and accountability mechanisms.
<i>Requirements</i>	Anonymity, Authorization, Confidentiality, Privacy
<i>Asset</i>	User Identities, User Data.
<i>Threats</i>	Threats related to IP spoofing, Threats related to user sensitive data (e.g. patients' medical file)
<i>Vulnerabilities</i>	Weak identification and authorization mechanisms, identity spoofing
<i>Related Patterns</i>	Accountability, Authorization.

Table 8: The Anonymization Security Pattern

<i>Pattern Name</i>	<i>Network Protection</i>
<i>Overview</i>	The Network Protection pattern is used to monitor which network ports of the e-health application are used, in order to block connections to unrelated ports and to other applications and/or OS services. Furthermore, it is used to keep track of network addresses that are sources of hacking attempts and other mischief. Any requests originating from an address on the blacklist are simply ignored.
<i>Problem</i>	In a network-based environment (especially internet-based), where anonymous access is possible, there is little that can prevent an attacker from brazenly attacking a system and/or an application with a variety of tools that are free and easy to use.
<i>Solution</i>	The Network Protection pattern is based on techniques that maintain a list of specific network ports that the e-health application should use. When an attacker tries to open a connection with another port, his request should be ignored without allowing use of other services of the application. On the other hand, the specific pattern requires techniques that maintain a blacklist of network addresses that have exhibited inappropriate behavior. When a request is received from a blacklisted address, it will simply be rejected
<i>Requirements</i>	Confidentiality, Integrity, Availability, Accountability
<i>Asset</i>	e-health Application, Operating System, e-health Servers
<i>Threats</i>	Well known network, internet related threats/attacks, users identities, user data
<i>Vulnerabilities</i>	Application bugs, inappropriate network configuration, inadequately controlled and/or monitored network connections
<i>Related Patterns</i>	No related patterns have been identified

Table 9: The Network Protection Security Pattern

<i>Pattern Name</i>	<i>Logging-Auditing</i>
<i>Overview</i>	E-health applications offer a variety of capabilities for logging events that are of interest to administrators and other users. If used properly, these logs can help ensure user accountability and provide warnings of possible security violations.
<i>Problem</i>	Some of the events that occur during the use of an e-health application are of particular interest. Recording specific information about events that have occurred creates records that allow the system to be debugged, monitored for security events, and measured for performance.
<i>Solution</i>	Every major component is responsible for logging events that it considers noteworthy. Some of these will be tagged as security-relevant events, others will not. E-health applications will typically deliver these events in some non-standard format to permanent storage, using one or more predefined log files.
<i>Requirements</i>	Accountability (Non repudiation)
<i>Asset</i>	e-health Application, Operating System
<i>Threats</i>	Threats related to application (indirect relationship)
<i>Vulnerabilities</i>	No related vulnerabilities have been identified
<i>Related Patterns</i>	No related patterns have been identified

Table 10: The Logging - Auditing Security Pattern

4. A security ontology for designing security patterns

4.1 Developing the ontology

Security patterns are used by developers to fulfill the security and privacy requirements of an e-health system. It should be noted that security patterns, such as the ones presented in the previous section, can be developed by security experts for different applications and can be applied in different contexts. The design of the security patterns follows the development of a security ontology; this is depicted in Figure 2. The development of the security ontology was carried out in the following phases:

During the first phase, a set of questions (called competency questions) was determined. These are loosely structured questions, indicating the type of answers and information we would like to receive when using the ontology. For the purposes of this work, the focus when building the ontology has been the area of e-health applications. This process enabled us to identify the important concepts within the e-health domain and the corresponding terms. The latter served as the basis for the formation of ontology classes (second phase). During the first phase, a large number of relevant terms was identified and recorded. Based on their relevancy to the e-health domain a subset of them was selected to form the ontology classes; other terms formed the properties of the classes; some terms were excluded as irrelevant. The terms used as ontology classes are the

following: Stakeholder, Objective, Threat, Countermeasure, Asset, Vulnerability, Deliberate attack, Security Pattern and Security Pattern Context. The third phase involved drawing the relations among the ontology classes and deciding upon their hierarchy. To reach this decision two approaches could be used: a top-down approach, where general concepts are included first and are later specialized; a bottom-up approach which suggests that specific classes are defined first and are then grouped into general concepts. We used a combination of the two approaches for designing the ontology. Finally, in the fourth phase we provided value types and allowed values or cardinality for the class properties (called slots) and the slot properties (called facets); this process is called instantiation of the ontology. These four phases were repeated several times. Following each iteration, the ontology was validated by means of the aforementioned competency questions; iterations ended only when the system could provide valid answers for these questions.

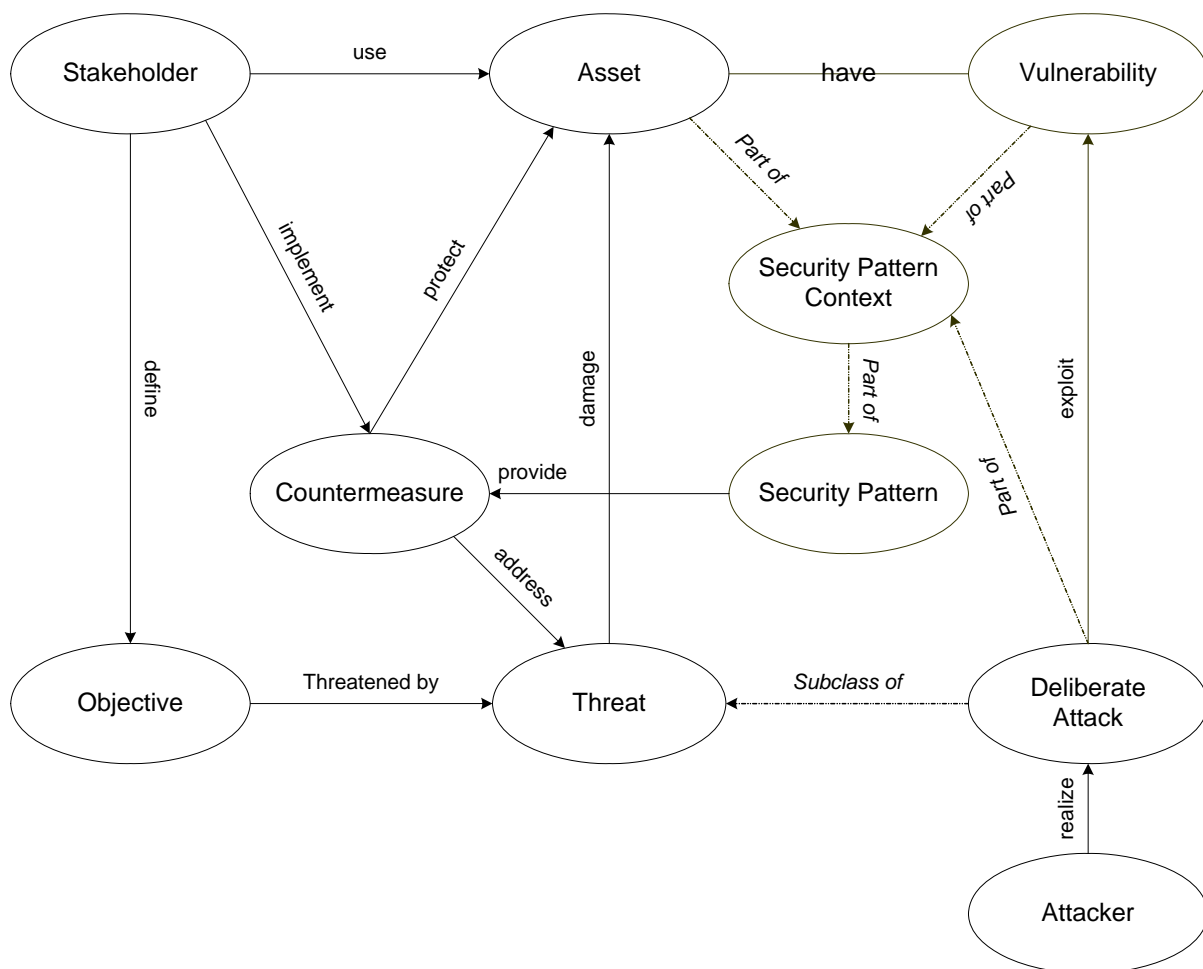


Figure 2. The security ontology.

In the ontology presented above, the concept of a Security Pattern is a representation of the security patterns and is connected with the concept of Countermeasures with a *provide* relationship: each security pattern provides a specific set of countermeasures. In practice, each security pattern is matched with a set of countermeasures during the ontology instantiation. A Security Pattern Context is defined as a set of Asset, Vulnerability and Deliberate Attack triplets. In this way, one can start from the generic security objectives, find the Security Pattern Contexts that match them and, thus, choose specific Security Patterns. In this way, the high level security requirements and objectives can be fulfilled by implementing the respective countermeasures.

4.2 Ontology development and validation

The tools used for developing and querying the security ontology were Protégé and Racer. The Protégé Ontology Editor has a modular design and itself provides only basic functionality; numerous plug-ins exist, depending on the task in hand. We chose the Protégé plug-in, which targets OWL¹ and RDF² ontologies. Racer is an inference engine that can be used for query answering over RDF documents. Racer was used to statically check our ontology for inconsistencies, and for submitting queries in order to verify its validity. The queries were expressed in the new Racer Query Language (nRQL). nRQL is a description logic query language for retrieving individuals from an A-box (a set of assertions about individuals) according to specific conditions. It allows the use of variables within queries that are bound against those A-box individuals that satisfy these conditions. The communication between Protégé and Racer was made possible through the nRQL Tab plug-in.

An indicative set of nRQL queries with their answers is provided below, illustrating what the security ontology can do for a developer involved in an e-health development project. These questions should not be regarded as exhaustive, but as indicative of what the ontology can deal with and reason about. Each question is first expressed formally as an nRQL query; then the result of executing this query is presented, and furthermore, where appropriate, the result is commented upon, or justified. These nRQL questions have allowed us to check that the ontology contained no inconsistencies.

¹ The OWL Web Ontology Language is designed for use by applications that need to process the content of information instead of just presenting information to humans. OWL facilitates greater machine interpretability of Web content than that supported by XML, RDF, and RDF Schema (RDF-S) by providing additional vocabulary along with formal semantics.

² The Resource Description Framework (RDF) integrates a variety of applications from library catalogs and world-wide directories to syndication and aggregation of news, software, and content to personal collections of music, photos, and events using XML as interchange syntax. The RDF specifications provide a lightweight ontology system to support the exchange of knowledge on the Web.

Q1. What are the typical objectives of an e-health system?

nRQL Query: (retrieve (?obj) (?obj |Objective|))
nRQL Result: (((?OBJ |Data_Confidentiality|))
((?OBJ |Availability|))
((?OBJ |Data_Integrity|))
((?OBJ |User_Eligibility|))
((?OBJ |User_Accountability|))
((?OBJ |User_Non_Repudiation|))
((?OBJ |Accuracy|)))

Q2. What threats might damage the medical data of a patient?

nRQL Query: (retrieve (?threat) (!Medical_Data| ?threat |damaged_by|))
nRQL Result: (((?THREAT |Impersonation|))
((?THREAT |Malicious_Code|))
((?THREAT |User_Error|))
((?THREAT |OS_Bugs|))
((?THREAT |Application_Bugs|))
((?THREAT |Terminal_Highjack|)))

Q3. What countermeasures protect the medical data of a patient?

nRQL Query: (retrieve (?cm) (!Medical_Data| ?cm |protected_by|))
nRQL Result: (((?CM |Encryption|))
((?CM |Access_Control|))
((?CM |Certificates|))
((?CM |Intrusion_Detection|))
((?CM |Malicious_SW_Detection|)))

Q4. What countermeasures address threats that are realized by vandals?

nRQL Query: (retrieve (?cm) (and (?cm ?threat |address|) (!Vandal| ?threat |realizes|)))
nRQL Result: (((?CM |Access_Control|))
((?CM |Firewall|))
((?CM |Backup_Policy|))
((?CM |OS_Security_Updates|))
((?CM |Intrusion_Detection|)))

Q5. What assets are confidential in an e-health system?

nRQL Query: (retrieve (?asset) (and (!Confidentiality| ?threat |is_threatened_by|) (?asset ?threat |damaged_by|)))
nRQL Result: (((?ASSET |Medical_Data|))
((?ASSET |Personel_Data|))
((?ASSET |Cryptographic_Keys|)))

5. Related work

Identifying and accommodating security requirements for applications have been research issues for quite a long time now. A detailed review of security design methods for information systems was first presented in [Baskerville \(1993\)](#). [Siponen \(2005\)](#) extended Baskerville's work by updating the list of security design and development methods and by providing an in-depth analysis and classification scheme ([Siponen 2005](#)). Siponen's analysis underlines the difficulty to integrate security into the overall information system development process as a major drawback of modern information system security development methods. Siponen also argues that increased attention should be paid to the integration aspect of methods and a shift to more socio-technical and social approaches should be made as a means to counter this problem.

5.1. The Object-Oriented approach

The object-oriented programming paradigm is defined by two basic properties: code is organized in units that have a single point of entry and a single point of exit – i.e. functions and procedures – and data are coupled with the functions/procedures that operate on them. One of the most widely used methodologies for the development of object-oriented software is the Rational Unified Process (RUP) ([Jacobson et al. 1999](#)). The RUP prescribes the construction of models, i.e. of semantically rich diagrams of the software that is being developed, which are used to communicate its requirements, architecture and design. These models are compiled in the Unified Modelling Language (UML). UML is a modelling and specification language that allows for the construction of diagrams of three kinds: Structural diagrams, Behaviour diagrams, and Model Management diagrams.

Since UML became a de facto industry standard, many efforts have been made to extend it -hence the RUP methodology as well- so as to include security design. Such an extension of the UML can be done mainly in two ways: through the use of profiles that extend Core UML – UML's defining meta-model – or at the very level of meta-models themselves by actually defining a new modelling language. UMLsec is an exemplary result of the former option whereas SecureUML is an exemplary result of the latter.

UMLsec is a standard language extension that was developed in order to allow the incorporation of security related information in UML diagrams ([Jurjens 2001](#)). It supports mechanisms for the verification of fulfilment of security requirements. However, it does not provide step-by-step instructions for reaching this end. The security requirements that can be expressed and validated using UMLsec include confidentiality, integrity, secure information exchange and access control. The major UML diagrams that UMLsec builds upon are the following ([Stevens and Pooley 2000](#)):

- Class diagrams, that are used to assure that information exchange satisfies the security level stated in the requirements.

- State Chart diagrams that are used to avoid covert information paths between higher and lower security level entities.
- Interaction diagrams that are used to verify secure interaction between entities.
- Deployment diagrams that are used to deal with the security of the physical layer.

SecureUML is a security modelling language that was proposed by [Basin et al. \(2003\)](#). Its syntax is defined by a MOF meta-model – MOF is a standard for defining meta-models. This underlying syntax is called “abstract syntax” so as to differentiate it from the “concrete syntax”, a notation used to create visual models with SecureUML just like with UML. Its semantics are based on an extension of the RBAC model. SecureUML was used by [Basin et al. \(2003\)](#) in combination with a UML-based process design language in order to demonstrate their broader concept of Model Driven Security ([Basin et al. 2003](#)). Model Driven Security is an approach stemming from Object Management Group’s (OMG) Model Driven ArchitectureTM (MDATM) ([Frankel 2003](#)). Just like MDATM, the purpose of Model Driven Security is to increase the quality of complex software systems through the creation of high-level system models and the consequent automatic generation of system architectures. In order to customize the MDATM paradigm for security, [Basin et al. \(2003\)](#) propose the combination of a security modelling language, like SecureUML, with a system design language through the use of a formal dialect that identifies primitives of the system design language as SecureUML resources. Model Driven Security allows the automatic generation of security architectures from models that are written in the combined language described above.

5.2. The Aspect-Oriented approach

Aspect-oriented programming is used to overcome a significant weakness of other programming paradigms (e.g. object-oriented programming), namely their inability to isolate the implementation of certain requirements—which in the aspect-oriented context are referred to as “concerns”—in a single unit of code (e.g. one class). Such concerns are called *crosscutting concerns*. Security requirements can sometimes be typical cases of crosscutting concerns. For example, the requirement for authorization must take place at each entry point of an application’s code. Even if only one such point lacks the appropriate authorization check, the application has a security hole. Ideally, the implementation of such a requirement should be isolated in a single unit of code, so that by verifying the correctness of this unit of code one could place a great deal of trust in the implementation as a whole.

An aspect-oriented framework for incorporating security requirements into software is the Acegi Security System. Acegi is built on top of Spring, the well known Java framework. Spring does not implement its aspect-oriented features on the language level but through the use of dynamic byte code generation, thereby leaving the Java language

intact. Acegi uses these aspect-oriented features to offer support for *authentication* and *authorization* services.

5.3. The Ontology-Driven approach

Raskin et al. (2001) advocate an ontological semantic approach to information system security design. Both the approach and its resources, the ontology and lexicons, are borrowed from the field of natural language processing and are adjusted to the needs of the security domain. This approach pursues the following goals: (i) the inclusion of natural language data sources as an integral part of the overall data sources in information security applications, and (ii) the formal specification of the information security community know-how for the support of routine and time-efficient measures to prevent and counteract computer attacks.

5.4. Security patterns

Security patterns represent standard solutions to common security requirements. They are an effective method for cataloguing and reusing existing security knowledge, as well as documenting software with security requirements. Security patterns can be thought of as specialized design patterns. In this respect, they can be documented and classified in a similar fashion as the one described by the “Gang of Four” (Gamma et al. 1995), that is documentation may fall under the following headings: pattern name and classification, intent, “also known as”, motivation, applicability, structure, participants, collaboration, consequences, implementation, sample code, known uses and related patterns. Security Patterns can be classified into creational, structural and behavioural patterns.

5.5. The Tropos approach

Mouratidis et al. (2003) have presented extensions to the well-known Tropos ontology to enable it to model security issues of agent-based systems. They have introduced the concept of security constraints that allow functional, non-functional and security requirements to be defined together, yet being clearly distinguished. They argue that their work makes it easy to identify security requirements at the early requirements stage and propagate them until the implementation stage.

5.6. A Reuse-Based approach

A research effort that is very close to our approach is the one made by Sindre et al (2003), who provide a reuse-based methodology for misuse case analysis. Their methodology comprises two main processes: the *development for reuse* and the *development with reuse*. The “development for reuse” process actually describes the methodology for choosing which development artefacts should be reused and also how their storage should be carried out. The latter includes guidelines for the construction and organization of appropriate repositories. In our approach security patterns are also reused and we provide guidelines for the construction of appropriate pattern repositories as well.

The “development with reuse” process describes an activity diagram with five steps (Identify Assets, Determine Security Goals, Specify Threats, Analyze Risks, and Specify Requirements). The authors analyze how steps 3 and 5 can be reused; they also suggest that the first two steps could also be reusable ones. Assets, Security Objectives, and Threats are also some of the main classes in the ontology we developed within our framework.

5.7. Developer guidelines and checklists

Developer guidelines and checklists are common in software development. While they do not guide developers in a step-by-step manner, they are nevertheless effective in helping them to avoid certain common pitfalls. For example, a security checklist of a large software company could include the following guidelines:

- Input validation.
- Simple design.
- Reuse code that has been proven to be secure.
- Defensive coding.
- Principle of least privilege.
- No security by obscurity.
- Passwords not transmitted in plaintext.
- Secure weakest link.

In order for such checklists to be as effective as possible, their context must be clear, their effectiveness must be continually evaluated and their content frequently renewed to reflect the changing environment.

5.8 Discussion

The approach described in this paper aims to exploit accumulated knowledge and expertise in the field of security requirements for the benefit of application developers. This knowledge-based approach is in line with the call for “increased attention to integration”, as put forth by [Siponen \(2005\)](#) since the use of patterns allows security solutions to be incorporated in the application development process.

Compared to other approaches, that have been described in the previous paragraphs, the use of security patterns coupled with the development of a security ontology provides better flexibility, as compared to the formal approach proposed by [Raskin et al. \(2001\)](#). Furthermore, this approach can provide solutions for all types of security requirements that may be relevant to an e-health application. The Acegi system, for example, is oriented towards addressing security requirements related solely to authentication and authorization.

In general, the majority of secure system development methods suffer from inadequate integration with system design methods. The approach described in this paper provides the follows features:

- It captures the knowledge of security experts and other system stakeholders and aims to use it to address the needs of the software developer. Other approaches, such as those proposed in (Jurjens 2001) and (Raskin et al. 2001) are meant to be used by security experts, not software developers.
- It employs an ontology to model the related concepts and the relationships among them. The developed ontology includes concepts, such as the security patterns, at a higher abstraction level, than that of other approaches (e.g. (Raskin et al. 2001) or (Mouratidis et al. 2003). Therefore, it allows more effective reusability and integration with existing approaches.
- The security ontology can be instantiated in different contexts, besides the e-health domain described in this paper. It can therefore model details that a more generic ontology, such as the one proposed in (Raskin et al. 2001), cannot capture.
- It is not limited in context, unlike approaches such as the one proposed in (Mouratidis et al. 2003), that can only be applied to agent-based systems.
- It can be employed for searching among possible solutions for the one that best fits the context, unlike approaches such as the one proposed by (Jurjens 2001) that are utilized to validate an already chosen solution.

6. Conclusions

The aim of this paper is to illustrate the way security patterns can facilitate the process of identifying security requirements and of selecting the designated security methods to address them. We have developed and validated an ontology that includes the major related concepts and the relationships that connect them. Based on this ontology, we have designed and developed a set of security patterns that address a subset of these requirements for applications that provide e-health services. The elements comprising the security patterns have also been decided on the basis of the developed security ontology.

Not all requirements have been mapped to the identified security requirements; this is an objective for future work. Moreover, further elaboration is needed with regard to the context of the security patterns.

7. References

Alexander C., Ishikawa S., Silverstein M., Jacobson M., Fiksdahl-King I., and Angel S. (1977), A Pattern Language, Oxford University Press.

Basin D., Doser J., and Lodderstedt T. (2003), "Model driven security for process-oriented systems", in Proceedings of the Eighth ACM Symposium on Access Control Models and Technologies (SACMAT '03), Como, Italy, ACM Press.

Baskerville R. (1993), "Information Systems Security Design Methods: Implications for Information Systems Development", ACM Computing Survey, 25(4): 375-414.

European Committee (EC) for Standardization (2003) Health informatics - International transfer of personal health data covered by the EU data protection directive - High level security policy, European Committee for Standardization, EN-14484.

Frankel S. (2003) Model Driven ArchitectureTM: Applying MDATM to Enterprise Computing, John Wiley & Sons.

Schumacher M., Fernandez B.E, Hybertson D., Buschmann F., Peter Sommerlad P., Security Patterns: Integrating Security and Systems Engineering, Wiley Series in Software Design Patterns, 2006.

Gamma E., Helm R., Johnson R., and Vlissides J. (1995), Design Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley.

Gritzalis S., Iliadis J., Gritzalis D., Spinellis D., and Katsikas S. (1999), "Developing Secure Web-based Medical Applications", Medical Informatics journal, Vol.24, No.1, pp.75-90, March 1999, Cambridge University Press - Taylor & Francis Publics.

Jacobson I., Booch G., and Rumbaugh J. (1999), The Unified Software Development Process, Addison-Wesley.

Jurjens J. (2001), "Towards development of secure systems using UMLsec", Lecture Notes in Computer Science, 2029:187.

Mouratidis H., Giorgini P., and Manson G. (2003), "An Ontology for Modelling Security: The Tropos Project", in Proceedings of the KES 2003 Invited Session Ontology and Multi-Agent Systems Design (OMASD'03), 2003, University of Oxford, United Kingdom.

Official Journal of the European Communities (1995) "Directive 95/46/EC of the European Parliament and of the Council of Europe of 24 October 1995, on the protection of individuals with regard to the processing of personal data and on the free movement of such data", Official Journal of the European Communities, Number L281/31, 23 November 1995.

Raskin V., Hempelmann C., Triezenberg K., and Nirenburg S. (2001) "Ontology in Information Security: A Useful Theoretical Foundation and Methodological Tool", In Proceedings of the New Security Paradigms Workshop, V. Raskin and C. F. Hempelmann (Eds.), 2001, New York, USA, ACM.

Sindre G., Firesmith D. G., and Opdahl A. L. (2003) "A Reuse-Based Approach to Determining Security Requirements", In Proceedings of the 9th International Workshop on Requirements

Engineering: Foundation for Software Quality (REFSQ'03), June 2003, Klagenfurt/Velden, Austria.

Siponen M. (2005), "Analysis of modern IS security development approaches: towards the next generation of social and adaptable ISS methods", *Information and Organization*, 15(4): 339-375.

Stevens P. and Pooley R., *Using UML*, 2000, Addison-Wesley.