Guan, H., Yang, H. and Wang, J. (2016) 'An ontology-based approach to security pattern selection.' *International Journal of Automation and Computing*, 13 (2): 168-182.

The final publication is available at Springer via http://dx.doi.org/10.1007/s11633-016-0950-1

An Ontology-based Approach to Security Pattern Selection

Hui Guan[1] Hongji Yang[2] Jun Wang[1]

[1]School of Computer Science and Technology, Shenyang University of Chemical Technology, Shenyang, 110142, China

[2] Centre for Creative Computing, Bath Spa University, SN130RP, UK

Abstract: Usually, the security requirements are addressed by abstracting the security problems arising in a specific context and providing a well proven solution to them. Security patterns incorporating proven security expertise solution to the recurring security problems have been widely accepted by the community of security engineering. The fundamental challenge for using security patterns to satisfy security requirements is the lack of defined syntax, which makes it impossible to ask meaningful questions and get semantically meaningful answers. Therefore, this paper presents an ontological approach to facilitating security knowledge mapping from security requirements to their corresponding solutions - security patterns. Ontologies have been developed using OWL and then incorporated into a security pattern search engine which enables sophisticated search and retrieval of security patterns using the proposed algorithm. Applying the introduced approach allows security novices to reuse security expertise to develop security software system.

Keywords: Security pattern, ontology, security requirement, risk analysis, security engineering.

1 Introduction

Experience shows that it is difficult to design a system that can fulfill the specific security requirements by simply integrating security mechanism and be error free at the same time, even for a small system. Security expertise tends to be valuable in such circumstances. However, such security expert knowledge is not always available for ordinary software developers. What's more, with software systems getting larger and more complicated, the situation of software security is getting even worse.

Inspired by design patterns, security patterns incorporate the security expertise to solve the recurring security problems in the specific contexts. For security novices, security patterns represent security best practices which are a convenient way to design secure and reusable software systems [1].They document basic mechanisms, processes or approaches which provide ways to safeguard CIA features of data [2].

In this paper, the security problems arising in legacy systems are addressed through using security patterns. Patterns are well-known solutions to recurring problems that arise in specific contexts and specify generic schemes with well-defined properties. Pattern writers have introduced many collections of security patterns recently. In [3], 415 published security patterns have been surveyed, of course, the number of existing security patterns is not limited to this. However, there are some features missing so as to impede the benefit of taking advantage of security patterns. One of the most fundamental features is how to find the "right" from the existing security patterns to solve the given specific security problem.

It is not possible to get right and meaningful answers automatically because of no syntax defined in security patterns [1]. Therefore, a framework for semantic description and management of security patterns via defining proper security ontology is developed in this paper. The fundamental idea is to ease the searching of "right" security patterns with the help of ontology technique. Security patterns can be described based on the proposed security core ontology which describes security patterns semantically and precisely. Therefore, sophisticated retrieval and search of security patterns are enabled.

2 Related work

The increasing importance of security in organizations leads to much research focusing on the inclusion of security concerns in the software development lifecycle. In this section, a review of recent studies on the security pattern selection is presented firstly with special regard to security pattern organisation and classification. Then, a brief review of existing ontologies in the information security domain is given.

2.1 Review of security pattern selection

The increasing number of patterns and similar security patterns appearing in the literature with different names makes it necessary to develop classifications of security patterns. A classification organises patterns into groups of patterns that share one or many properties such as the application domain or a particular purpose. Various security pattern classification approaches have been proposed since Gamma et al. introduced the first classification of security patterns (GoF patterns) [4].

Heyman et al. [5] classified 220 security patterns into three categories: guidelines, process and core patterns. Design guidelines described by Viega and McGraw in [6] were used to compare 8 security patterns by Cheng et al. in [7]. They extended their classification based on access types of security patterns and thereby classified patterns in terms of application levels: network-level, host-level and

This work was supported by Research Project of Education department of Liaoning province (China) (No. L2013156), National Scholarship (No.201208210386), and Key Industry Problem Plan of Liaoning (No. 2012219001).

application-level. Kienzle et al. [8, 9] classified security patterns into two broad categories, i.e., structural and procedural. Another broad classification of security patterns was made by Blakley et al. [10], in which two broad categories of security patterns were made: available patterns and protected patterns. Halkidis et al. [11] examined the evolution feature of security patterns by comparing the patterns derived from [10]. Laverdiere et al. [12] proposed a six sigma method to classify the 12 common security patterns from [7] and [11]. Hafiz et al [13, 14] proposed a multi-dimension classification scheme taking consideration of security CIA features, application context, security wheel, McCumber cube, STRIDE threat modelling, and hierarchical classification. The relationships used in their work are similar to the dependencies among security problem patterns suggested by Hatebur et al. [15].

In [16], an ontological interface for software developers to select security patterns was proposed. The proposed interface contains a mapping between security requirements on the one side and threat models, security bugs, security errors on the other side taking into consideration their contexts of applicability. However, they did not give the design and implementation of their ontology.

Montero et al. [17] proposed a semantic representation for domain specific patterns based on domain knowledge. The representation was used as an underlying basis for complementing the textual description of pattern using semantic annotations. However, their approach did not fulfill its objective in selecting the appropriate set of security patterns.

It will benefit software developers by providing a means to request security patterns through unnecessary classified security terms. It is meaningful if the classification approaches for security pattern are transparent to software developers, especially those security novices. From this point of view, the approaches mentioned above cannot satisfy such kind of need.

2.2 Review of security ontology

Understanding user's security concerns plays an important role in software development since security as one of the NFRs becomes more and more important in the success of modern software. Ontology can facilitate the process as it is regarded as a good approach to systematically classifying and categorising various security concerns as well as related security countermeasures. Therefore, security ontology is considered as an important research area within the security engineering by more and more researchers.

Some security ontologies have been proposed for general concepts in information security domain. Herzog et al. [18] proposed an OWL-based ontology of information security on the basis of a book named Principle of Information Security [19]. They endeavoured to design an extensible ontology for the information security domain that covers the whole general concepts. For a similar goal, Fenz et al. [20] proposed an security ontology that covers a more broader spectrum with 500 concepts. Their ontology is then applied to quantified risk assessment by integrating ISO/IEC 27001 standard ontology.

Compared with the above ontologies, the followings describe specified aspects of security. Velasco et al.[21] proposed an ontology framework for representing and reusing security requirements based on risk analysis. Security risk ontology and security requirements ontology were developed based on the requirement engineering standards and implemented using OWL. Tsoumas et al. [22] defined a security ontology of risk analysis based on the standards to provide security acquisition and knowledge management. The security ontology acts as a container for the security requirements.

Schumacher et al. [1] proposed a security ontology to maintain the security pattern repository with a theoretical security pattern search engine. However, only top level concepts were introduced in their ontology, which is too abstract to be applied to the specific context.

Dobson et al. [23] proposed an ontology in dependability domain. Denker et al. [24] developed several ontologies for security annotations of agents and web services. Karyda et al. [25] proposed a security ontology using OWL with which to develop secure e-applications. Security patterns were defined to capture security expertise to support secure application development.

Although several ontologies in information security domain exist, none of them has been proposed to map the security knowledge from security requirements to their corresponding solutions - security patterns. This paper uses ontology as a vehicle for managing different security requirements, security patterns and their mapping relationships.

3 Framework of the approach

An overview of the proposed security enhancement framework is shown in Fig. 1.

The main emphasis will be given to security pattern to guide software developers in their effort to fulfil security requirements through the design and implementation of security solutions so as to provide reliable security services.

In the proposed approach, security requirements are elicited by risk analysis approach derived from our previous work [26]and formalised as a list in which elements related to security requirement (SR) are represented as columns containing asset (*A*), threat (*T*), security feature (*SF*) and priority (*P*). The meaning of these elements in the same row is, for a given asset *A*, one or more threats *Ts* may threaten *A* by violating one or more security feature *SFs*. *P* is quantified by using the security vector approach derived from our previous work [27]. Therefore, each software requirement can be fulfilled in a sequence according to the value of *P* during software development.

Based on the security requirements specification, a pattern searching method is designed for automatic identification and retrieval of the most suitable security patterns that fulfil the given security requirement with the aid of the proposed security ontology inference. In order to achieve this goal, both security requirements and security patterns are semantically described and stored.

Security requirements are represented semantically with OWL to enable automatic mapping to their solution. Each element that makes up the security requirement is semantically described, categorised and organised into different abstraction levels. Take the element "threat", for instance. It will be classified into Application-level,

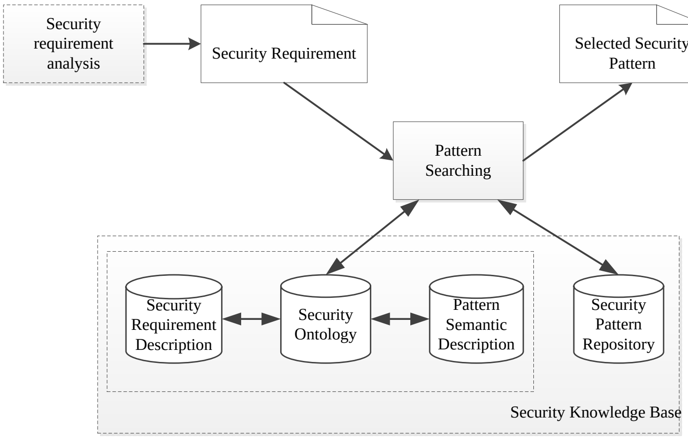

Security requirement analysis
Security Requirement
Selected Security Pattern
Security Pattern Integration
Pattern Searching
Security Requirement Description
Security Ontology
Pattern Semantic Description
Security Pattern Repository
Security Knowledge Base

Fig.1 Operational framework

Host-level and Network-level. Each category will be further organised into sub-categories.

Security patterns are semantically described with specific profiles and solutions for various contexts. The descriptions of security pattern include abstraction level, type of solution, applicability, context conditions and security properties provided by the pattern. A series semantic properties is defined to each pattern, such as “security attribute: Confidentiality”, “Deployed in design phase” …and so on. The incorporation of precise and rich semantic descriptions of the security patterns enables the use of automated reasoning mechanisms capable of searching proper patterns to fulfil the given security requirement.

Security patterns are formatted and stored in a repository to support the following security pattern integration. While the appropriate security pattern is found via the pattern search engine, corresponding security pattern document can be selected from the pattern repository and thereby be integrated into the system model.

Besides the semantic description of security requirements and security patterns, mapping algorithms and inference rules as parts of security ontology are developed and stored to form a security knowledge base together with the security pattern repository.

Fig. 2 shows the structure of the proposed security knowledge base. Basically, the structure of the security knowledge base is similar to a tree structure for storing security related information that helps to reveal and organise the security relevant features, and for relating these properties to fundamental security requirements. It consists of two sub repositories, security ontology base and security pattern base. Security ontology base is used to store the established ontology including concepts and relationships while security pattern base is the repository to store and organise the common security patterns for further processing. Considering the reusability of the stored security relevant information, the information is expressed in a generalised way and focuses on the abstract level.

Finally, selected security patterns will be adapted, instantiated and integrated in the system design model to be implemented by software developers. Therefore, security features can be incorporated to protect the system against security attacks.

4 Security requirement elicitation

Security requirements represent the types and levels when attempts to protect the assets to meet security policy

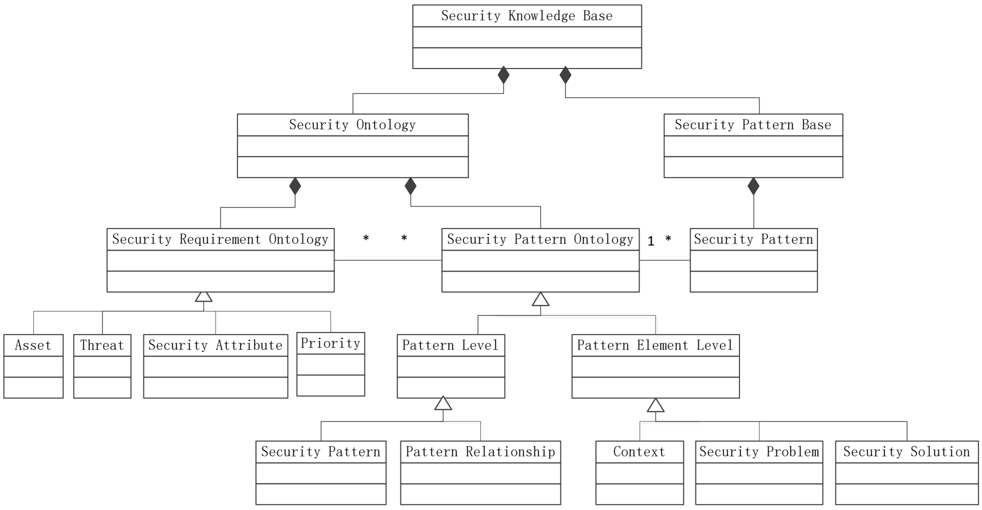


Fig.2 Class diagram for the meta-model of security knowledge base

[28]. A complete and consistent group of security requirements can be produced by using an elicitation method. Specially, security requirements are identified by risk analysis—“the systematic use of information to identify sources and to estimate the risk” [29].

Usually, most functional requirements are specified as what must happen, while security requirements are stated in terms of what must not be allowed to happen. After the risk analysis, assets can be enumerated with criticality level, threats threatening the assets can be elicited with severity risk level, security features that would be violated by potential threats on assets can be analysed, and priority level representing the developing order of the security requirements can be computed.

It is impossible to develop a completely secure system because of the budget, deadline, and resources needed for the development and the emerging new kinds of attacks, even if it could be done, the usability and efficiency of the developed system may be decreased. Thus, developing secure systems is about trade-offs and it is quite a challenge to find a balance point. Prioritising of each elicited security requirement and incorporating user’s security objectives play a key role when facing such a dilemma.

The criticality of each asset has to be evaluated, which implies a criterion for the security threshold of an asset is decided according to not only the impact value but also the risk for the asset, including likelihood and impact. Therefore, analysing the threat and vulnerability of a system in order to evaluate the risk is required. Specially, analysis of the threats threatening to the system is used as s a means of identifying why the assets need protection. In addition, the vulnerability of the system is detected and analysed in order to understand what weakness exists in the system that can be exploited by the threats. This is the process of security requirement elicitation. The outcome of this process will be a list of security requirements with priorities representing their criticalities to the system. Table 1 shows an example output of security requirement elicitation.

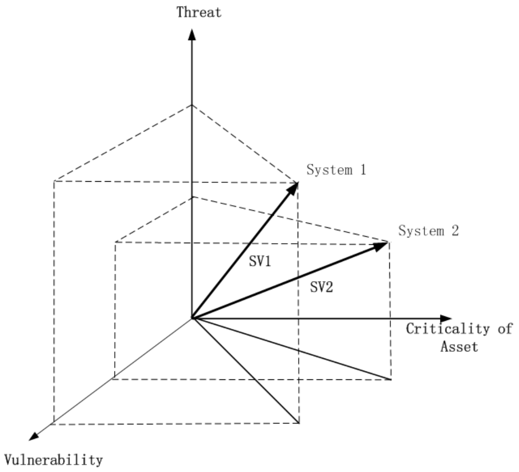


Fig. 3 Security vectors[27]

The security level of a software system can be illustrated in Fig. 3 and be quantified by using the security vector approach *SV* <*A*, *T*, *V*> in (1) proposed in the previous work [27]. It can be used as the priority order of security requirements when system designers develop security aspects or countermeasures to fulfil them. Table 1 shows an example of security requirement format in this paper.

$$\mathrm{SV} = \sqrt{A^2 \times W_\mathrm{a} + T^2 \times W_\mathrm{t} + V^2 \times W_\mathrm{v}} \qquad (1)$$

where W_a is the weight of asset *A*, W_t is the weight of threat *T*, and W_v the weight of vulnerability *V*. Security factors including asset, threat and vulnerability are quantified and treated as the elements of the security vectors *SV*.

5 Security pattern

Security patterns incorporate proven security expertise solutions to the recurring security problems. Usually, the security requirements are addressed by abstracting the security problems arising in a specific context and providing a well proven solution to them [30]. The ability of security patterns to mitigate and stop security threats can be found in [11, 31] where security patterns incorporated into the system could contribute to the improvement of system’s security level [32].

It should be noted that security patterns can be designed and developed by security experts for different kinds of problem solving and be applied to different contexts. For example, they can be abstract higher level architectural patterns that specify how to resolve a security problem architecturally, or they can be even more abstract patterns that depict the process to secure software development, or they can be defensive design level patterns describing how the detailed security artefacts can be created [30].

5.1 Security pattern format

The documentation of security patterns were originally built by Yoder et al. [33] in 1997. Seven architectural security patterns are presented and structured using the formats in POSA [34] or GoF [4] which are generic schemes for describing design patterns in the architecture level.

The format is composed of several elements shown as follows [7]:

- Intent: description of goal and issues the pattern addresses;
- Context: description of situations or environment in which the pattern is used;
- Problem: description of the problem that this pattern solves;
- Description: description of the scenarios that illustrate the design problem;
- Solution: description of the solution to the problem;
- Consequences: description of the trade-offs and results when this pattern is used;
- Forces: description of constraints that should be considered when the pattern is applied.
- Known uses: description of the patterns use found in real systems;
- Related patterns: description of the related patterns that use this pattern as a reference.

In the view of pattern format, pattern authors can describe all sections which they consider of importance. Therefore, for all the elements in a security pattern, just Problem and Context elements are useful while searching proper security patterns from a security point of view.

The structure of security patterns adopted in this paper is based on the traditional design patterns. They have an expressive name, an application context, problem to be solved and a solution to that problem.

Therefore, security pattern is represented as a 3-tuple <Context, Problem, Solution> where:

- Context defines the conditions and situation in which the pattern is applicable

Time and location are usually regarded as important characteristics of context in the security domain. Time relates to when a security problem occurs and the location specifies at which level of the system infrastructure a security problem occurs. In terms of software domain, typical example of the time within a context is software life cycle phases which are analysis, design, implementation, integration, and location where the operation occurs usually expressed as application, host and network [1].

- Problem defines the vulnerable aspect of an asset

The problem field of a security pattern is important for software developers to determine whether a security pattern is appropriate for their situation. This field defines the security problems that occur in the specific contexts and can be solved by the security pattern. A security problem occurs whenever a system is unprotected or is protected insufficiently against abuse or misuse. Generally speaking, the security problem can be some kinds of threats which cause possible danger or damage when someone or something violates security policies.

- Solution defines the scheme that solves the security problem which occurs in the security context

Security solution is a group of countermeasures to be applied in the system in order to mitigate the security risk. It is meaningful that at least one security countermeasure is implemented to keep the system invulnerable for each threat.

5.2 Security pattern organisation

A significant number of security patterns have been proposed since the first effort in 1977 by Yoder et al. [33]. A security pattern may address more than one security feature, for example, Authentication pattern can protect both confidentiality and integrity security features. At the same time, for a specific security property, there may be more than one security pattern to address it. It is a many-to-many situation. Additionally, security patterns may be organised by different parameters from abstract to more specific. Hence, it is difficult to find the “right” security patterns for solving a particular security problem without a proper classification scheme of security patterns [14]. A suitable classification scheme not only contributes to efficient information storage and retrieval, but also benefits both pattern navigators and pattern miners.

In this section, on the basis of several existing classification frameworks, an efficient classification framework for security patterns has been described to facilitate finding the proper security patterns according to the elicited security requirements. As the security requirement is based on threat modelling and asset analysis, the properties of threat and asset will be considered as the factors for selecting security patterns. The proposed classification scheme is based on multiple aspects of the relevant information.

- Lifecycle Stage.

While most of the security patterns take the form derived from design patterns, not all security patterns are dedicated to design phase. Therefore, classification on the lifecycle stages is meaningful for organising security patterns ordered on the dichotomy of beginning and end, which are: Analysis, Architecture, Design, Implementation, and Deployment.

- Architectural Layer.

Layer provides another useful dimension, since problems and their solutions in different layers of the architecture differ, yet all are important. Roughly, the architecture has been divided with an ordering from low to high level of abstraction. The following distinctions are used as the architecture layers, which are: Data, Application, System, and Network.

- Application Context.

The structure of the system is usually taken into consideration as another classification factor to partition the security patterns according to which part of the system they are trying to protect [14]. The security of a system is analysed from three levels: core security, perimeter security and exterior security. The core security deals with the security implementation within the system while the perimeter security focuses on security related issues at the system entry points, such as authorisation, authentication and security. The exterior security considers protecting data during transmission and securing communication protocols.

- Domain Specific

Application domain can provide an important differentiator or a filter to narrow the field of applicable knowledge [35]. Some security pattern solutions are specific to a particular domain or application type. This dimension is an exception in that it does not have a dichotomy or ordering—the space is freely defined. Pattern designers can create patterns for their own domain as a form of knowledge capture. After examining the existing security pattern, several example domains are provided in this paper: Ubiquitous computing, Distributed computing, Web and J2EE, Embedded system, Operating system, Service oriented architecture, SCADA (supervisory control and data acquisition), and not limited to this coverage.

- Threat Type.

The classification scheme based on threat modelling is more intuitive because it uses the security problems that the patterns solve. Security architects use threat modelling to identify and prioritise a system’s security threat which makes the prioritisation of the mitigation effort possible. STRIDE [36] is one of the widely used models to classify threats according to different sources. It is the English acronym of the following six threat types [36]:

  - Spoofing is someone or something masquerades to be legitimate and valid.
  - Tampering is data interfered or modified during network communication.

    - Repudiation is the situation that user denies performing a certain action which could be illegal and harmful.
    - Information disclosure is when an unauthorised user gets access to confidential information, which he or she is not supposed to have access to.
    - Denial of service is basically when a service is brought down intentionally or unintentionally, resulting in the disruption of normal services for legitimate users.
    - Elevation of privilege is when an unauthorised user gets higher privilege access from the one he or she was supposed to have, which might result in access to restricted information, or might apply dangerous tasks.
- Security Concerns

Software patterns are usually chosen by software developers with a particular goal in mind. Developers tend to view security in terms of software requirements rather than taking the perspective of an attack. Therefore, it is necessary to apply security goals or concerns to classify the security patterns. This metric is more straight and easier understood to software designer to select proper security patterns in the security design. In this paper, the most frequently used security concerns are listed as: Access control, Authentication, Confidentiality, Integrity, Availability, Accountability, and Non-repudiation.

For better visualisation, Table 2 summarises the classification scheme for security patterns.

6 Security ontology

An ontology, in the field of knowledge representation, is most often defined as “a representation of a conceptualisation” [37]. A more detailed description of ontology is that “it is a formal representation of the entities and relationships which exist in some domains, it should also represent a shared conceptualisation in order to meet any useful purpose” [23]. Ontologies are useful for representing and inter-relating many kinds of knowledge. In 2003, Marc Donner urged the necessity of having good security ontologies. He argued that too much security terminology is vaguely defined, thus it becomes difficult to communicate between colleagues and, worse, confusing to deal with the people we try to serve: “What the field needs is an ontology – a set of descriptions of the most important concepts and the relationships among them. A great ontology will help us report incidents more effectively, share data and information across organisations, and discuss issues among ourselves” [38].

The advantages of applying ontology technology into the information security domain are specified in [39] from three viewpoints: (1) ontologies can eliminate the ambiguity of items to a properties list and organise information in a systematic way at detailed level; (2) ontological technology can induce the modularity which can be used by other approaches, for example, to detect some new features by establishing relations among different measurements; and (3) an ontological approach has the ability to forecast security problems by providing inference mechanisms.

The approach proposed in this paper can be summarised by the following points. The security patterns for software engineering are created to document the knowledge of the experts in security field. These patterns are designed by using the ontology techniques that provide reusable and structured activities or solve security problems that may arise during the development of software systems. Moreover, due to the OWL representation, the security patterns are available in a machine readable format and it is expected to be automatically utilised in the system.

This section addresses the issue of fulfilling elicited security requirements. The approach uses ontologies as a tool for managing different security requirements and associating them with corresponding security solutions provided by security patterns.

The main goal is to provide a security ontology based framework, which unifies the proposed methods in security evolution for legacy systems. The ontology “knows” which threats threaten which assets, and which security patterns could lower the probability of occurrence in which contexts. It is meaningful for the software developer to find the appropriate security patterns by adopting an ontology based approach [16].

6.1 Overview of the proposed ontology

The proposed security ontology is designed to achieve the following goals:

- Describe risk relevant information especially security requirement information applicable to web application
- Design security pattern ontology at two abstraction levels
- Facilitate mapping security requirements to security patterns
- Provide the ability to annotate security related information to facilitate the security pattern selection
- Create reusable and easy to extend ontologies

The designed ontologies are supposed to be used by both the security pattern providers who design new security patterns and edit the corresponding ontology into the ontology base to express their security capabilities, and the security requirement requestors who have got security requirements to be fulfilled by security patterns. From the security requestor’s point of view, security requirements can be stated in terms of 4-tuple <Asset, Threat, Security Attribute, Priority>, which is elicited from the proposed risk assessment method in [27]. When it comes to the security pattern provider, the security capabilities are expressed in terms of security patterns which are organised as 3-tuple <Context, Security Problem, Security Solution>.

The proposed ontology has been developed by using OWL, which is a language based on RDF for processing web information by the computer rather than being read by people. OWL is the current recommendation of W3C (World Wide Web Consortium) for processing the content of web information. OWL is a part of semantic web and has three sublanguages, OWL Lite, OWL DL (including OWL Lite), and OWL Full (including OWL DL). Based on Description Logics, OWL-DL has been used to design the

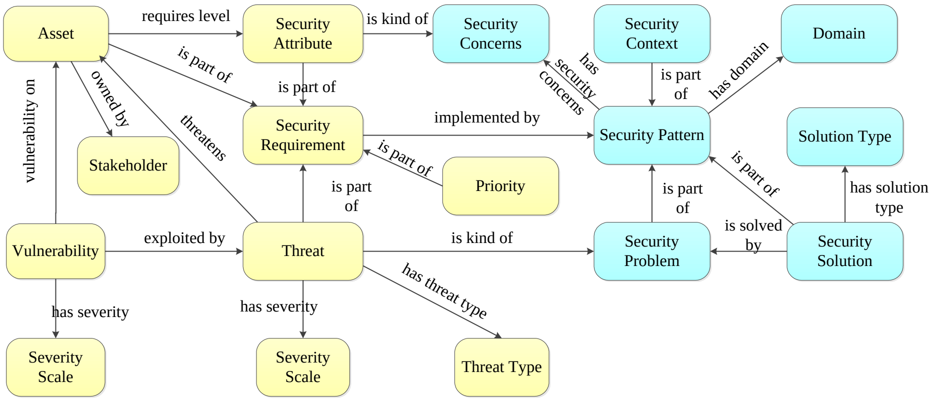


Fig. 4 Proposed security ontology top level concepts and relations

proposed ontology for its expressivity is suitable for the requirement and allows for complete reasoning by DL reasoner, for example, Racer, FaCT++ or Pellet.

The tools used for developing and querying the security ontologies are Protégé and FaCT++. The Protégé Ontology Editor (Protégé) provides the graphical interface for ontology designers to build OWL ontologies. However, the Protégé itself only provides editing function so that a reasoner (FaCT++, in this study) is required to check the consistency of the developed ontology.

6.2 Development of the proposed security ontology

Designing OWL ontology is not only defining a set of classes and properties, but also including a collection of restrictions and axioms. This ensures that the correct result can be inferred from the proposed ontology.

There are several methods to develop ontology. The method used in this paper is based on METHONTOLOGY [40]. The development of security ontology is carried out in the following phases:

- Define questions. A collection of questions within the domain is defined to indicate what kind of answers and information are expected by using the ontology. The questions are informal and loosely structure as any forms. Some important concepts can be identified during this process, which can be termed as the basis when building ontology classes.
- Build classes. Based on the previous phase, a lot of relevant concepts and terms have been identified and recorded. They can be classified and selected according to their relevancy to the domain to form the classes, or properties of the proposed ontology.
- Build relationships. This process involves clarifying the relationships among the classes and defining the hierarchy. It is the process of adding axioms and restrictions to the ontology. Axiom is a set of assertions specifying what is true in the domain. It is used to connect classes and properties with some logical information about them. Restriction is a special kind of class description with which all individuals in that class will satisfy the restriction.
- Build ontology instance. This is the procedure to create instances of the classes, which refers to inserting the individual information or providing examples of each of the classes.
- Validate ontology. The competence questions built in the first phase can be used to validate the correctness of the proposed ontology.

The aforementioned phases have been repeated several times until the provided answers from the proposed ontology satisfy the competency questions.

To accomplish the automatic mapping between security requirements and security patterns, a security ontology is developed based on [20] and its top-level concepts and relations are shown in Fig. 4. It is composed of two subontologies: security requirement subontology (sr) and security pattern subontology (sp). The security requirement subontology consists of the core concepts: Asset (sr:asset), Threat (sr:threat), Vulnerability (sr:vulnerability), Attribute (sr:attribute), Priority (sr:priority). The security pattern subontology is composed of the core concepts: Security Context (sp:context), Security Problem (sp:problem), Security Solution (sp:solution). The concepts of sr:asset have been derived from [41], sr:vulnerability and sr:threat from [42], while security pattern subontology concepts are derived from [1].

6.3 Security requirements subontology

In our previous work [26, 27], the security requirement is identified by risk analysis, which is one of the sources to elicit security requirement. Consequently, the requirement ontology (Fig. 5) is developed with the concepts derived from the risk analysis using Protégé Editor. The meta-information associated with risk analysis (such as asset and threat) can be used to define axioms, constraints and rules that help to maintain the consistency of the proposed security requirement ontology.

Every security requirement is a description of which asset is threatened by which kind of threat by violating which security objective and to which severity extent. The properties defined in security requirement ontology are described below:

- Each requirement is characterised by a unique identifier and has been defined as Datatype property in OWL.

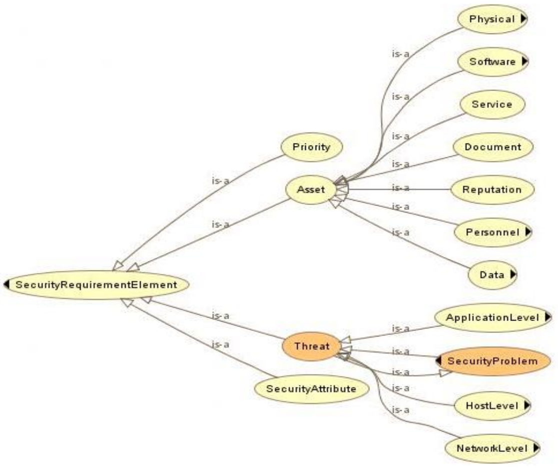


Fig. 5 Top level of security requirement ontology

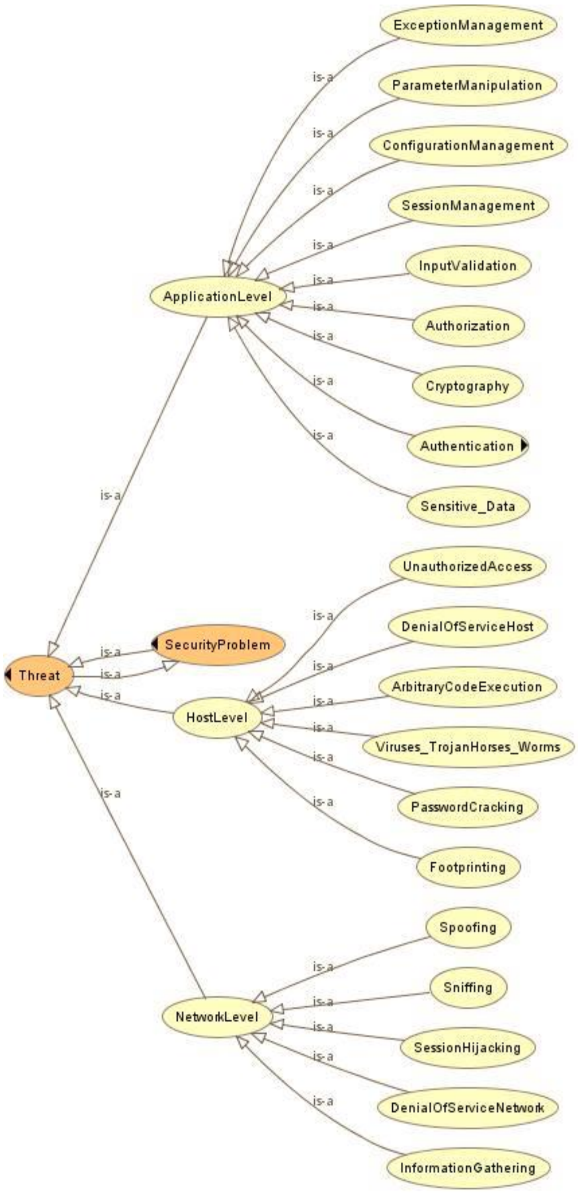


Fig. 6 Top level of threat ontology

- hasAsset: it represents the asset related to this requirement. It is defined as an object property with domain defined as class SecurityRequirementElement and range as class asset.
- isThreatenedBy: it represents possible threats that endanger the asset and then make the requirement unfulfilled. This property is represented by an object property and its range is the class Threat (Fig. 6) defined in this ontology. According to the risk analysis in [27], there are constraints of which threat can occur to which asset. Fig. 7 shows partial of the security ontology defined in OWL.
- hasSecurityAttribute: the features that make an asset valuable. There exist four types of security properties using an object property: “Confidentiality”, “Integrity”, “Availability” and “Accountability”.
- hasPriority: the value can be computed by using (1) taking asset criticality, threat severity and vulnerability severity scale into account and shows the order of development. Datatype property {“high”, “Medium”, “Low”}.

```
<owl:Class rdf:about="&Security;Asset">
          <rdfs:subClassOf
rdf:resource="&Security;SecurityRequirementElement"/>
</owl:Class>
< owl:Class rdf:about="&Security;Threat">
       <rdfs:subClassOf
rdf:resource="&Security;SecurityRequirementElement"/>
</ owl:Class>
   …
<owl:ObjectProperty rdf:about="&Security;isThreatenedBy">
      <rdfs:domain rdf:resource="&Security;Asset"/>
      <rdfs:range rdf:resource="&Security;SecurityProblem"/>
      <inverseOf rdf:resource="&Security;hasAsset"/>
</owl:ObjectProperty>
   …
<NamedIndividual rdf:about="&Security;DataTampering">
      <rdf:type rdf:resource="&Security;Authorization"/>
      <rdf:type rdf:resource="&Security;Sensitive_Data"/>
      <Security:residesOn
        rdf:resource="&Security;Application"/>
</NamedIndividual>
```

Fig. 7 Partial ontology definition in OWL

6.4 Security pattern subontology

As described in Section 4.1, the structure of the security pattern is a 3-tuple <Context, Problem, Solution> from the security point of view. Moreover, there are relationships among security patterns.

Fig. 8 illustrates of security pattern subonotolgy which is based on [1]. The main properties of the pattern subontology are shown below:

- Security patterns are characterised by a unique identifier and a text description. Both have been defined in OWL as Datatype properties.
- hasContext: it represents the situation in which the security problem occurs and is defined as object property. The range of it is subclass SecurityContext. Two subproperties are hasLayer

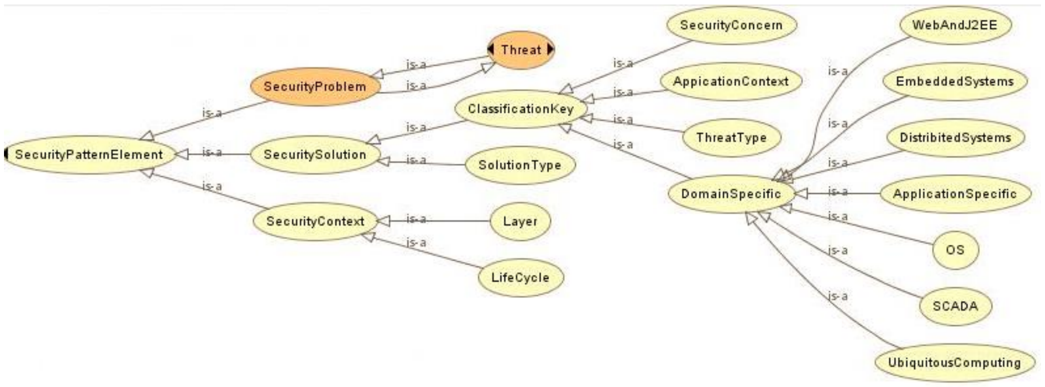


Fig. 8 Top level of security pattern ontology

and hasLifeCycle, whose ranges are Layer and LifeCycle, respectively.

- hasProblem: it represents the security problem that occurs in such a security context and is defined as object property. The range of it is subclass SecurityProblem and an axiom is added as equivalent as subclass Threat in Security Requirement subontology.
- hasSolution: it represents the security solution to the security problem that occurs in the given security context.
- hasThreatType: it represents the problem type classified according to threats whose domain is SecurityPattern and range is ThreatType.
- hasSecurityConcerns: it represents the security features the security pattern holds.
- hasDomain: application domain the security pattern serves. It is defined as object property whose domain is SecurityPattern and range is Domain.
- requires: it represents the Require relationship between security patterns. It is added as object property with the range being SecurityPattern.
- isSpecialisedBy: it represents the Specialise relationship between security patterns. It is added as object property with the range being SecurityPattern.

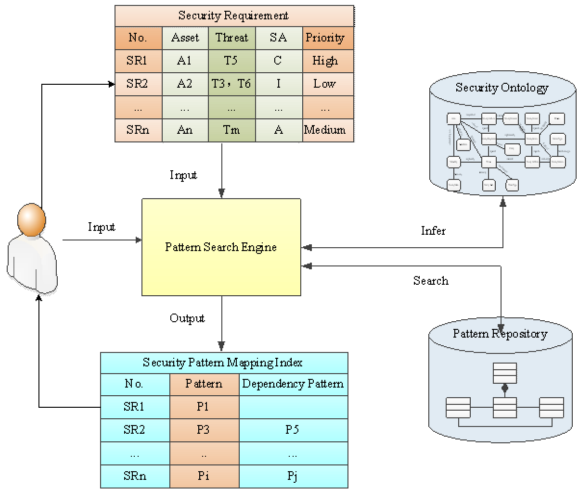
7 Security pattern search engine

Since this study aims to support the security pattern selection process provided that the security requirements have been elicited, a security pattern search engine (Fig. 9) is designed to facilitate the process and therefore to validate the proposed security ontology.

In this case, 32 security patterns are selected from the published literatures and form the pattern repository which can be extended as needed. An example of the proposed security pattern repository is partially shown in Table 3. Patterns in the repository are organised and labeled using the proposed classification scheme.

The pattern search engine is composed of four functions and can be implemented by incorporating OWL API:

- Input function. An input function receives the user’s required security requirement or takes the set of security requirements as input.
- Infer function. An infer function infers the developed security ontology to find the security patterns according to the user input by using OWL API. The core of infer function is the algorithms realising the mapping.
- Search function. A search function will search the security pattern repository according to the mapping result of infer function and returns the development specification of the selected patterns which can be used by developer.
- Output function. An output function returns the mapping index between security requirement and mitigation security patterns.



Security Requirement

No.	Asset	Threat	SA	Priority
SR1	A1	T5	C	High
SR2	A2	T3，T6	I	Low
...	...	...	...	...
SRn	An	Tm	A	Medium

Security Pattern Mapping Index

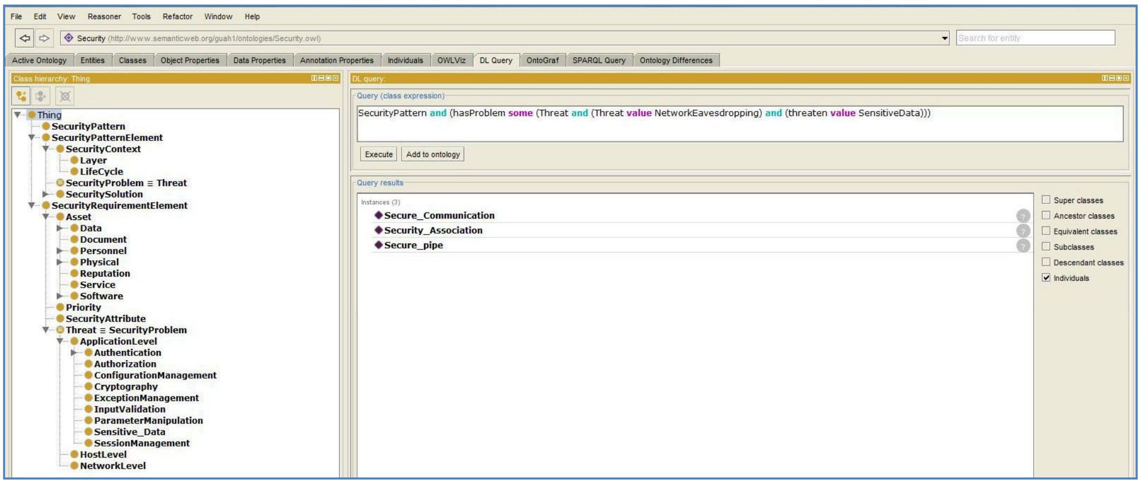
No.	Pattern	Dependency Pattern
SR1	P1	
SR2	P3	P5
...	..	...
SRn	Pi	Pj

Fig. 9 Pattern selection process

The key part of the pattern search engine is some algorithms that match the security patterns with required security requirements until either there are no more security requirements existing, or there are no more security patterns which can be matched with them.

7.1 Algorithm

In order to extract the corresponding results from the proposed security ontology, the Protégé OWL API can be used to encode the competency questions in the algorithm structure. The OWL API is a Java application interface and reference implementation for creating, manipulating and serialising OWL ontologies. In the following, a representative algorithm is given in a pseudo code format to show how the search engine performs the infer function.



Query (class expression)

SecurityPattern and (hasProblem some (Threat and (Threat value NetworkEavesdropping) and (threaten value SensitiveData)))

Query results

Instances (3)

- Secure_Communication
- Security_Association
- Secure_pipe

☐ Super classes
☐ Ancestor classes
☐ Equivalent classes
☐ Subclasses
☐ Descendant classes
☑ Individuals

Fig. 10 Example of query result in Protégé editor

By incorporating OWL API, Algorithm 1 is used to search the security patterns which can mitigate the threats threatening the given asset by violating the given security attributes in a given domain. In the GetRelated(*x*, *y*) function, *x* is a given concept, while *y* is a relation (also called object property in Protégé OWL). The GetRelated(*x*, *y*) function returns a collection of concepts which are related with x via y. The GetInstances(*x*) function returns a collection of instances (also called individuals in Protégé OWL) belonging to concept *x*.

7.2 Evaluation

The security pattern search engine aims to provide the inferring and searching capability with an interactive interface. The kernel of engine is the proper ontology definition and matching algorithm.

Result of the pattern searching process is a data set comprising the selected security patterns, which is then evaluated by security expert. Evaluation of the result is the process of evaluating the efficiency of the proposed security ontology.

Usually, the system developers come up with competency questions to validate the ontology. The questions are designed as indicative of what the ontology can handle and reason about rather than as exhaustive as possible. In this paper, each of the questions is firstly expressed formally as a DL-query, which is a query language that can be used to query RDF and OWL-DL ontologies, and then the query results are presented with comments in appropriated place. One of the examples is illustrated in Fig. 10 showing the evaluation result while using the proposed ontology to process the security pattern searching.

Q1: Which threats threaten the integrity attribute of internal data assets in the network layer?

DL Query*: Threat and (threaten some (Asset and (Asset value InternalData) and (SecurityAttribute value Integrity)) and (resideOn value Network))*

DL Result*: Spoofing*
Session Hijacking

Q2: Which security patterns protect the sensitive data against the threat of network eavesdropping?

DL Query*: SecurityPattern and (hasProblem some (Threat and (Threat value NetworkEavesdropping) and (threaten value SensitiveData)))*

DL Result*: Secure Pipe*
Secure Communication
Secure Association

Q3: Which security patterns can be used in Web and J2EE domain to address the SQL injection threat?

DL Query*: SecurityPattern and (hasDomain value WebAndJ2EE) and (hasProblem and (Threat value SQLInjection))*

DL Result*: Input validator*

Due to the high degree of complexity, it is inefficient to answer all of the competency questions using simple ontology queries. However, it illustrates the ability of ontology to answer such complex questions.

8 Conclusions

Based on our previous work of security requirement elicitation, this paper promotes the application of security pattern to the secure software development. Security patterns make it possible for security novices to integrate security expertise into their development. However, the number of security patterns and their different representation forms make it difficult to select the "right" patterns for fulfilling a given security requirement.

In this paper, an ontological approach is proposed to manage security requirements, security patterns and the mapping relationships among them. The ontology has been developed using formal method and implemented in OWL. The ontology facilitates security knowledge mapping from security requirements to security patterns. The definition of proposed ontology is based on security requirement derived from the previous work [27] and knowledge of security pattern from [2, 4, 30]. Moreover, a prototype capable of searching security patterns is designed by processing the knowledge contained in the proposed ontology.

The proposed approach is novel and unique. It smooths the transferring from security requirements to secure architecture by using security patterns. It combines security requirements, the pattern approach and ontology paradigm in order to improve the application of security patterns to security engineering domain.

Future directions for this approach will focus on two main areas. One is the extension of the requirement ontology using widely accepted standards, such as OCTAVE or ISO/IEC 27001. The other area is the implementation of the prototype system, in which the expert systems might be used to improve the selection of security patterns.

References

[1] M. Schumacher. *Security Engineering with Patterns: Origins, Theoretical Models, and New Applications*. Berlin, Germany: Springer-Verlag Berlin Heidelberg, 2003.

[2] M. Schumacher, E. Fernandez-Buglioni, D. Hybertson, F. Buschmann and P. Sommerlad. *Security Patterns: Integrating Security and Systems Engineering*. UK: John Wiley & Sons, 2006.

[3] M. Bunke, R. Koschke and K. Sohr. Organizing security patterns related to security and pattern recognition requirements. *International Journal on Advances in Security*, vol. 5, no. 1 and 2, pp. 46-67, 2012.

[4] E. Gamma, R. Helm, R. Johnson and J. Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Boston, USA: Pearson Education, 1994.

[5] T. Heyman, K. Yskout, R. Scandariato and W. Joosen. An analysis of the security patterns landscape. In *Proceedings of the Third International Workshop on Software Engineering for Secure Systems*, IEEE Computer Society, Minneapolis, MN, pp. 3, 2007.

[6] J. Viega and G. McGraw. *Building Secure Software: How to Avoid Security Problems the Right Way*. Boston, USA: Addison-Wesley Professional, 2001.

[7] B. H. Cheng, S. Konrad, L. A. Campbell and R. Wassermann. Using security patterns to model and analyse security requirements. *IEEE Workshop on Requirements for High Assurance Systems*, pp. 13-22, 2003.

[8] D. M. Kienzle and M. C. Elder. Final Technical Report: Security Patterns for Web Application Development, University of Virginia, USA, 2002.

[9] D. M. Kienzle, M. C. Elder, D. Tyree and J. Edwards-Hewitt. Security Patterns Repository Version 1.0. *DARPA, Washington DC*, 2002.

[10] B. Blakley and C. Heath. Security Design Patterns Technical Guide - Version 1, the Open Group, UK, 2004.

[11] S. Halkidis, A. Chatzigeorgiou and G. Stephanides. A qualitative evaluation of security patterns. *Information and Communications Security*: Springer Berlin Heidelberg, pp. 132-144, 2004.

[12] M. A. Laverdiere, A. Mourad, A. Hanna and M. Debbabi. Security design patterns: survey and evaluation. *Conference on Electrical and Computer Engineering(CCECE '06)* pp. 1605-1608, 2006.

[13] M. Hafiz and R. E. Johnson. Security Patterns and Their Classification Schemes, University of Illinois at Urbana-Champaign Department of Computer Science, USA, 2006.

[14] M. Hafiz, P. Adamczyk and R. E. Johnson. Organising security patterns. *IEEE Software*, vol. 24, no. 4, pp. 52-60, 2007.

[15] D. Hatebur, M. Heisel and H. Schmidt. Analysis and component-based realization of security requirements. In *Third International Conference on Availability, Reliability and Security(ARES'08)* IEEE CS Press, Barcelona, pp. 195-203, 2008.

[16] P. El Khoury, A. Mokhtari, E. Coquery and M. S. Hacid. An ontological interface for software developers to select security patterns. In *19th International Workshop on Database and Expert Systems Application(DEXA '08)*, Turin, pp. 297-301, 2008.

[17] S. Montero, P. Díaz and I. Aedo. A Semantic Representation for Domain-Specific Patterns. *Metainformatics*, U. Wiil Ed., Germany: Springer Berlin Heidelberg, pp. 129-140, 2005.

[18] A. Herzog, N. Shahmehri and C. Duma. An ontology of information security. *International Journal of Information Security and Privacy (IJISP)*, vol. 1, no. 4, pp. 1-23, 2007.

[19] M. Whitman and H. Mattord. *Principles of Information Security (2nd Edition)*. Boston: Course Technology, 2005.

[20] S. Fenz and A. Ekelhart. Formalizing information security knowledge. In *Proceedings of the 4th International Symposium on Information, Computer, and Communications Security*, ACM, United States, pp. 183-194, 2009.

[21] J. L. Velasco, R. Valencia-García, J. T.

Fernández-Breis and A. Toval. Modelling reusable security requirements based on an ontology framework. *Journal of Research and Practice in Information Technology*, vol. 41, no. 2, pp. 119, 2009.

[22] B. Tsoumas and D. Gritzalis. Towards an ontology-based security management. In *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, Vienna, pp. 985-992, 2006.

[23] G. Dobson and P. Sawyer. Revisiting ontology-based requirements engineering in the age of the semantic web. In *Proceedings of the International Seminar on Dependable Requirements Engineering of Computerised Systems at NPPs*, Halden, 2006.

[24] G. Denker, L. Kagal and T. Finin. Security in the semantic web using OWL. *Information Security Technical Report*, vol. 10, no. 1, pp. 51-58, 2005.

[25] M. Karyda et al. An ontology for secure e-government applications. In *The First International Conference on Availability, Reliability and Security(ARES 2006)*, IEEE, Vienna, Austria, pp. 5, 2006.

[26] H. Guan, W. Chen, L. Liu and H. Yang. Environment-driven threats elicitation for web applications. *Agent and Multi-Agent Systems: Technologies and Applications*: Springer, pp. 291-300, 2011.

[27] H. Guan, W. Chen, L. Liu and H. Yang. Estimating security risk for web applications using security vectors. *Journal of Computers*, vol. 23, no. 1, pp. 54-69, 2012.

[28] J. Lasheras, R. Valencia-García, J. T. Fernández-Breis and A. Toval. Modelling reusable security requirements based on an ontology framework. *Journal of Research & Practice in Information Technology*, vol. 41, no. 2, pp. 119-133, 2009.

[29] ISO/IEC 17799-272002. *Code of Practice for Information Security Management*, 2005.

[30] C. Steel, R. Nagappan and R. Lai. *Core Security Patterns: Best Practices and Strategies for J2EE, Web Services, and Identity Management*: Prentice-Hall, 2005.

[31] S. T. Halkidis and E. Chatzigeorgiou. A practical evaluation of security patterns. In *Proceedings of the International Conference on Artificial Intelligence and Digital Communications*, pp. 1-8, 2006.

[32] E. B. Fernandez, N. Yoshioka, H. Washizaki and M. VanHilst. Measuring the level of security introduced by security patterns. In *2010 International Conference on Availability, Reliability and Security*, Krakow, pp. 565-568, 2010.

[33] J. Yoder and J. Barcalow. Architectural patterns for enabling application security. In *Proceedings of the 4th Conference on Pattern Languages of Programming (PLoP '97)*, USA, 1998.

[34] F. Bushmann, R. Meunier, H. Rohnert, P. Sommerlad and M. Stal. *Pattern-Oriented Software Architecture: A System of Patterns*: John Wiley&Sons, 1996.

[35] M. VanHilst, E. B. Fernandez and F. Braz. A multi-dimensional classification for users of security patterns. *Journal of Research & Practice in Information Technology*, vol. 41, no. 2, pp. 87-98, 2009.

[36] F. Swiderski and W. Snyder. *Threat Modeling*: Microsoft Press, 2009.

[37] T. R. Gruber. Toward principles for the design of ontologies used for knowledge sharing. *International Journal of Human-Computer Studies*, vol. 43, no. 5, pp. 907-928, 1995.

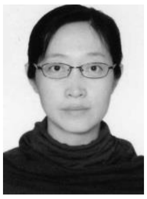
[38] M. Donner. Toward a security ontology. *IEEE Security & Privacy*, vol. 1, no. 3, pp. 6-7, 2003.

[39] V. Raskin, C. F. Hempelmann, K. E. Triezenberg and S. Nirenburg. Ontology in information security: a useful theoretical foundation and methodological tool. *Proceedings of the 2001 Workshop on New Security Paradigms*, pp. 53-59, 2001.

[40] A. Gomez-Perez, M. Fernández-López and O. Corcho. *Ontological Engineering*: Springer Heidelberg, 2004.

[41] BSI. BS7799 - Code of Practice for Information Security Management, 1999.

[42] J. D. Meier, A. Mackman, S. Vasireddy, M. Dunner, R. Escamila and A. Murukan. *Improving Web Application Security: Threats and Countermeasures*: Microsoft, 2003.



Hui Guan received her B.Sc. and M.Sc. degrees in computer science from the Shenyang Institute of Chemical Technology, China, in 2000 and 2006, respectively, and her PhD in software engineering in 2014 from De Montfort University, England. She is currently an associated professor at School of Computer Science and Technology, Shenyang University of Chemical Technology.

She has published 10 referred journal and conference papers. Her research interests include software security, model driven development and software evolution.

Dr. Guan received research financial support from Research Project of Education Department of Liaoning province (China) in 2010 and 2013, respectively.

E-mail: h.guan@syuct.edu.cn (Corresponding author, ORCID ID: orcid.org/0000-0002-2450-8568#sthash.foVfwfzz.dpuf)

Hong-Ji Yang obtained his BSc and MPhil in computer science in 1982 and 1985, respectively from Jilin University, China, and his PhD in computer science in 1994 from Durham University, England. He is currently a professor at the Centre of Creative Computing, Bath Spa University, England.

His research interests include software engineering and pervasive computing. He served as a program co-chair at IEEE International Conference on Software Maintenance 1999 (ICSM'99) and the program chair at IEEE Computer Software and Application Conference 2002 (COMPSAC'02).

Prof. Yang is an IEEE computer society golden core member, 2010.
E-mail: h.yang@bathspa.ac.uk

Jun Wang received the B.Sc. and M.Sc. degrees in computer science from the Shenyang Institute of Chemical Technology, PRC in 2001 and 2005, respectively, and the Ph.D. degree from Shenyang Institute of Automation of CAS, PRC in 2009. Currently, he is an associate professor, and a vice-dean in the School of Computer Science and Technology at Shenyang University of Chemical Technology, PRC. He leads the Network and Information Security Lab. In January 2010, he was invited as an academic visitor (including post-doctoral project as a post-doctor) at De Montfort University, UK, and he has been collaborating with Professors Yang, Zedan and Chen of that university.

His research interests include software reliability in distributed computing systems, wireless network, and the Internet of things.

E-mail:wj_software@hotmail.com

Table 1 Example of security requirements

SR No.	Asset	Threat	CIAA	Priority
SR1	User bank account	Sniffing	Confidential	High
SR2	User account	Cross-site Scripting	Confidential, Integrity	High
SR3	Place order	User Denies Performing an Operation	Accountability	Low
SR4	Display product	Denial of Service	Availability	Medium
SR5	Product Catalogue	Data tampering	Integrity	Medium

Table 2 Summary of the proposed multiple aspects classification scheme

Criteria	Classification						
Application	Core	Perimeter	Exterior				
Architectural	Data	Application	System	Network			
Lifecycle Stage	Analysis	Architecture	Design	Implementation	Deployment		
Domain Specific (Not limited)	Ubiquitous computing	Distributed computing	Web and J2EE	Embedded system	Operating system	SOA	
Threat Type	Spoofing	Tampering	Repudiation	Information disclosure	Denial of service	Elevation of privilege	
Security Concerns	Access control	Authentication	Confidentiality	Integrity	Availability	Accountability	Non-repudiation

Table 3 Example of security pattern repository organised by proposed classification scheme

Pattern Name	Application	Architectural	Lifecycle Stage	Domain Specific	Threat Type	Security Concerns
Audit Interceptor [30]	Core	Application	Design	Web and J2EE	Repudiation	Accounting
Authenticator [1]	Perimeter	Application	Design	ALL	Spoofing	Authentication
Authorisation [1]	Perimeter	Application	Architecture	ALL	Information Disclosure	Access Control
Checkpointed System [10]	Core	Application	Architecture	ALL	Tampering	Availability
Intercepting Validator [30]	Core	Data	Design	Web and J2EE	Spoofing	Integrity
Secure Logger[30]	Exterior	Data	Design	Web and J2EE	Tampering	Accountability Non-repudiation
Secure Pipe[30]	Exterior	Network	Design	Web and J2EE	Information Disclosure	Confidentiality

Algorithm 1 Security patterns searching

Input	A is the given asset SA is the given security attribute D is the given application domain
Output	SP is the security pattern array
Initialisation	SP=∅

```
procedure getAsset(A, SA, D) return SP
  1.  A ← given asset
  2.  SA← given security attribute
  3.  D← given domain
  4.  SP←Null
  5.  TL← GetRelated(A, sr:isThreatedBy)
  6.  for i←0 to TL.Length do
  7.  T← GetInstance(TL[i])
  8.  for j←0 to T.Length do
  9.  if T[j].sr:hasSecurityAttribute ==SA then
  10.     P← GetRelated(T[j],sp:isSolvedBy)
  11.     for k← 0 to P.Length do
  12.             PI ← GetInstance(P[k])
  13.             for m ← 0 to PI.Length do
  14.             if PI[m].sp:hasDomain= = D then
  15.                 if PI[m].sp:hasLayer = = T[j].sr:residesOn then
  16.                      PR← GetRealted(PI[m], sp:requires)
                              {*PR is the pattern set in which pattern is required by the exacted pattern
                              with "require" relation in security pattern subontology sp*}
  17.                          PS←GetRelated(PI[m], sp:isSpecialisedBy)
                              {*PS is the pattern set in which pattern specifies the exacted pattern with
                              "isSpecialisedBy" relation in security pattern subontology sp*}
  18.                         if PS.Length !=0 then
  19.                            for l← to PS.Length do
  20.                               SP.Add(PS[l])
  21.                              end for
  22.                          else
  23.                              SP.Add(P[m])
  24.                          end if
  25.                          if PR.Length !=0 then
  26.                            for n← to PR.Length do
  27.                              SP.Add(PR[n])
  28.                              Line 16 to Line 27 with PR[n] for PI[m]
  29.                             end for
  30.                           end if
  31.                     end if
  32.                 end if
  33.             end for
  34.         end for
  35.     end if
  36.   end for
  37. end for
  38. return T
  39. return SP
```