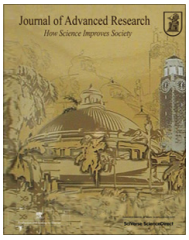
Journal of Advanced Research (2014) 5, 463–472

Cairo University

Journal of Advanced Research



ORIGINAL ARTICLE

Capturing security requirements for software systems

Hassan El-Hadary, Sherif El-Kassas *

Department of Computer Science & Engineering, The American University in Cairo, Egypt

ARTICLE INFO

Article history:
Received 6 October 2013
Received in revised form 1 March 2014
Accepted 3 March 2014
Available online 12 March 2014

Keywords:
Application security
Security requirements engineering
Security threat modeling
Problem frames

ABSTRACT

Security is often an afterthought during software development. Realizing security early, especially in the requirement phase, is important so that security problems can be tackled early enough before going further in the process and avoid rework. A more effective approach for security requirement engineering is needed to provide a more systematic way for eliciting adequate security requirements. This paper proposes a methodology for security requirement elicitation based on problem frames. The methodology aims at early integration of security with software development. The main goal of the methodology is to assist developers elicit adequate security requirements in a more systematic way during the requirement engineering process. A security catalog, based on the problem frames, is constructed in order to help identifying security requirements with the aid of previous security knowledge. Abuse frames are used to model threats while security problem frames are used to model security requirements. We have made use of evaluation criteria to evaluate the resulting security requirements concentrating on conflicts identification among requirements. We have shown that more complete security requirements can be elicited by such methodology in addition to the assistance offered to developers to elicit security requirements in a more systematic way.

Introduction

During the last decade, software systems security has become an increasingly growing concern due to the large number of incidents and attacks targeting software systems [1]. Attackers exploit software vulnerabilities and cause threats to the systems such as stealing sensitive information, manipulating data and causing denial of service. One of the grand challenges in information security is to develop tools and principles that allow construction of large-scale systems for important security critical applications such as e-banking systems and electronic voting systems [2].

Secure software development includes integrating security in different phases of the software development lifecycle (SDLC) such as requirements, design, implementation and testing. Early consideration for security in requirement phase helps in tackling security problems before further proceeding in the process and in turn avoid rework [3]. Several approaches have been proposed that upgrade previous requirement engineering approaches to let it support security such as goal oriented [4], agent oriented [5] and UML use case based [6].

Abbreviations: AF, abuse frame; SPF, security problem frame; PF, problem frame; SR, security requirement; AR, anti-requirement.

* Corresponding author. Tel.: +202 2615 2974.
E-mail addresses: sherif@aucegypt.edu (S. El-Kassas).

Peer review under responsibility of Cairo University.

2090-1232 © 2014 Production and hosting by Elsevier B.V. on behalf of Cairo University.
http://dx.doi.org/10.1016/j.jare.2014.03.001

In order to integrate security with requirement engineering, we have to consider security requirements. The basic task of security requirement engineering is to identify and document requirements needed for developing secure software system. Satisfying such *security requirements* should lead to more secure software system [7]. We adopted the definition that considers security requirements as constraints on the functionality of the system focusing on what should be achieved. We agree that the security requirements should be expressed as positive statements and not negative statements. Expressing requirements in such way can help in verifying its satisfaction [7]. Security requirements can be elicited by analyzing the assets to be protected and the threats from which these assets should be protected [8].

Security requirements need to be adequate as possible. They need to be explicit, precise, complete and non-conflicting with other requirements [4]. However, knowledge of security is a basic necessity prior to practicing security requirement engineering. The analyst should have background on how to identify and analyze the system assets, threats, vulnerabilities and requirements. One of the challenges for secure software systems development is to assist developers in performing security requirements engineering [9]. A more effective approach for security requirement engineering is needed to provide a more systematic way for eliciting adequate security requirements.

Problem frames [10] are means that can be used to reuse previous knowledge in modeling software problems in the requirement engineering process. Several approaches provide solutions to adapt security while following a problem frames based requirement engineering process such as abuse frames [11] and security problem frames [12]. Problem frames are used in different frameworks for identifying security requirements such as Haley's approaches [7,13]. However, we have a gap between such approaches. No integration is presented in the literature that bridges them together although they complement each other. This paper proposes a methodology for security requirement elicitation that provides a more systematic way for software developers in order to elicit adequate security requirements while following a problem frames based requirement engineering process. The methodology considers security while eliciting the requirements of software systems using problem frames. The main goal of the methodology is to assist developers to elicit adequate security requirements during the requirement engineering process with the aid of previous security knowledge. A security catalog, based on problem frames, is constructed for this purpose. The scope of the methodology is limited to the requirements phase in the SDLC, and is not intended to cover security through the entire SDLC.

This paper is organized as follows. Section "Related work" discusses related approaches for security requirement elicitation. Section "Methodology" presents our proposed methodology for security requirement elicitation. Section "Results and discussion" compares results of applying our methodology with two related methodologies. Section "Conclusion" summarizes our work and suggests areas for future work.

Related work

Different requirement engineering approaches are updated in order to consider security such as UML use cases [6,14,8], agent oriented [5,15], goal based [4,16] and problem frames based requirement engineering [7,12,17]. New models are introduced to represent threats that can be exploited by the attackers such as attack trees [18], misuse cases [6], anti-models [4] and abuse frames [11]. Moreover, threats classification and analysis techniques are introduced such as STRIDE and DREAD [19]. Thus, the approaches are updated to consider threats and elicit security requirements that mitigate such threats.

Moreover, reusing security knowledge is tackled in different approaches in order to assist software developers in eliciting security requirements in a systematic way. For example, security problem frames [12], misuse cases templates [3], and anti-models patterns [20] are used to form *generic* model based catalogs which are not specified for a particular application. Thus, the developer can make reuse of such generic models and templates during elaborating threats and security requirements.

Our methodology is mainly based on problem frames. In Section "Problem frames," we will cover problems frames and approaches that integrated security with problem frames.

Problem frames

Problem frames [10] are means that can be used in the requirement engineering process to describe software development problems. They can help in analyzing problems to be solved where interaction between the software and domains in the system context is described. Problem frames are useful in requirements engineering because they help in decomposing the system context into simpler subproblems which are mapped to well-known problem classes [21]. Thus, problem frames provide helpful means to reuse previous knowledge in modeling software problems including security related problems.

Different approaches provide solutions to integrate security while using problem frames based requirement engineering process. Abuse frames [11] and security problem frames [12] are means for modeling security problems. Moreover, problem frames are utilized in different frameworks for eliciting security requirements. For example, Haley's approaches [7,13] made use of problem frames in order to identify vulnerabilities and elicit security requirements.

Methodology

The proposed methodology aims at early integration of security with software development. It considers security while eliciting the requirements of software systems using problem frames. The methodology aims at identifying security requirements with the aid of previous security knowledge through constructing a security catalog for this purpose. The security catalog consists of problem frame models for *threats* and the corresponding *security requirements*. Threats are modeled using abuse frames while security requirements are modeled using security problem frames.

Section "The methodology steps" describes the methodology steps while giving examples for applying the methodology on a software banking system. Section "Methodology iterations and outputs" elaborates how the methodology iterates through its steps. Section "Security catalog" illustrates the structure and the contents of the security catalog used throughout the methodology.

The methodology steps

The methodology iterates through the following steps:

1. System modeling.
2. Assets identification.
3. Threats and vulnerabilities identification.
4. Security requirements elicitation.
5. Security requirements evaluation.

A flow chart for the methodology is shown in Fig. 1. We will apply the methodology on a simple software banking system to explain the steps:

Step 1: System modeling

In this step, we will use problem frames to model the problem context of the system and to decompose it into subproblems. The output of this step will be the problem context diagram in addition to the problem frame diagrams representing the functional requirements of the system. Fig. 2 represents the problem context diagram while Fig. 3 represents the problem frame diagram for a simple software bank system.

Step 2: Assets identification

After modeling the system, we will specify the assets in the system. We can follow the technique used in previous studies [7,13] where assets are identified by checking the *domains* of the subproblems constructed in the first step of the methodology. For example, if we are modeling a software banking system, we can have the domain *Account Information* in two subproblems: one for editing account information and another one for viewing account information. Such domain represents an asset because it requires preserving security concerns such as confidentiality, integrity, availability, accountability and authenticity.

Step 3: Threats and vulnerabilities identification

After identifying the assets of the system, we will identify the threats that can harm such assets. The threats will describe *what* the attacker can do in order to violate the security concerns of the system. We will search for threats in the

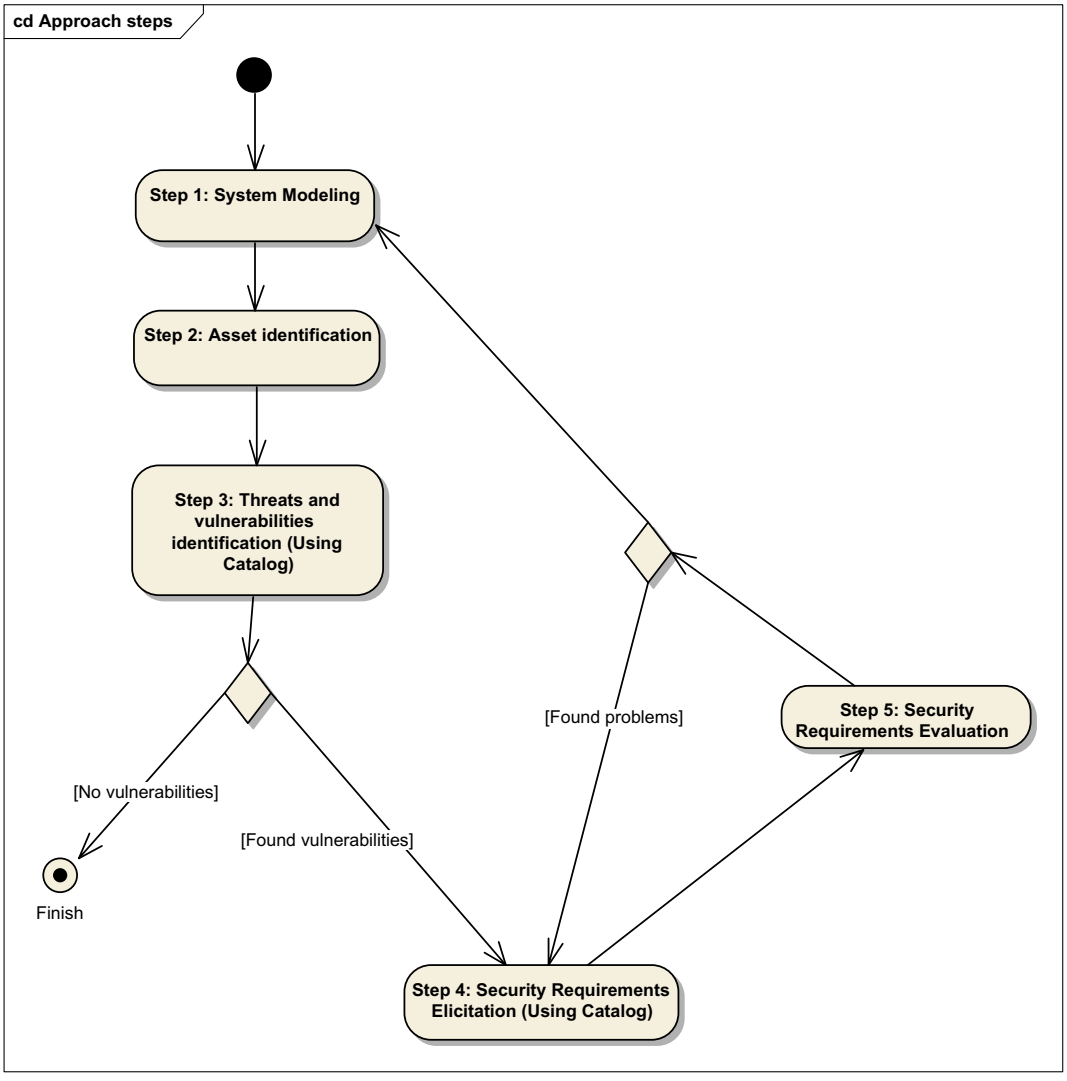


Fig. 1 Flow chart for the proposed methodology.

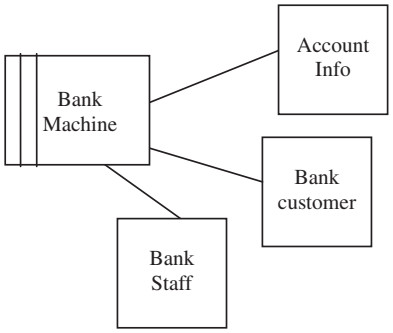
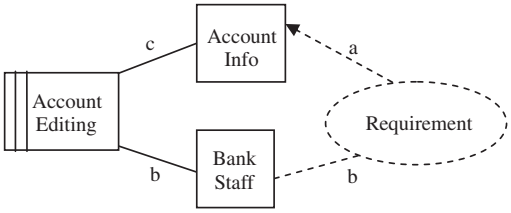


Fig. 2 Problem context diagram for a simple software bank system.



a: AI!{Content of account info}
b: BS!{EnterAccountNo,EnterAmount}
c:AEM!{update Amount} AI!{RetrieveAmount}

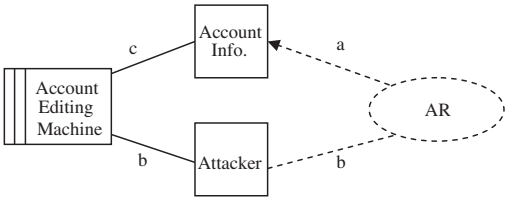
Requirement*: Bank operator edits account info by crediting funds*

Fig. 3 PF1: Crediting funds to account subproblem frame diagram [33].

security catalog. Such catalog is constructed from threats, modeled by abuse frames [11] and corresponding security requirements modeled by security problem frames [12]. The security catalog contents are illustrated in Section "Security catalog."

We will check whether any of the threats in the catalog can cause vulnerability in the system. A vulnerability is a weakness in the system that maybe exploited by an attacker [13]. Mainly, vulnerabilities are caused if the system allows the threats to occur. Such threats can cause harm to the system because they violate any of its security concerns (confidentiality, integrity, availability, accountability and authenticity). We will make use of Haley's approach [7,13] to identify vulnerabilities where threats are crosscut with functional requirements we have modeled in step 1. This is because threats are related to assets which are represented by domains in the subproblems. For example, the tampering data threat found in the security catalog is concerned with unauthorized modifications to stored data such as account information. After crosscutting such threat with the subproblem "PF1: Crediting funds to account" represented by the problem frame diagram in Fig. 3, we can find that the attacker can perform manipulation to account information which represents our asset. Such threat can cause a vulnerability because the current model of the subproblem does not include a mitigation for it.

After identifying threats that can cause vulnerabilities, we will instantiate the abuse frames [11] that model the discovered threats. Abuse frames will be instantiated by substituting the domains, phenomena and interfaces in order to meet the context of the system. For example, Fig. 4 shows the instantiated



a: AI!{Account amount modified}
c: AEM!{Modify account information}
b: ATT!{Execute commands to modify account info without authorization}

AR: *Attacker makes modifications to account information without authorization*

Fig. 4 AF1: Tampering account information abuse frame diagram.

abuse frame (AF) diagram "AF1: Tampering account information" that models the tampering threat affecting the account editing subproblem in a software banking system. In such diagram, the *Attacker* domain represents the malicious user who can exploit the vulnerability in the machine *Account Editing Machine*.

In some cases, we might need some design assumptions in order to identify if there exists any vulnerabilities. For example, we might need to know how the data are transferred from domain to another in order to identify whether the threat of disclosing transmitted data is applicable. If the designer announced that the transmission medium is encrypted and secure, we can ignore the threat.

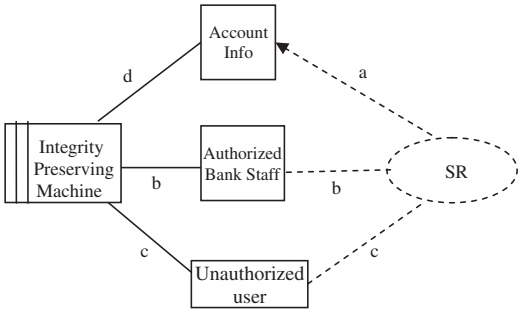
The output of step 3 will be composed of instantiated abuse frames or *abuse frame diagrams* modeling the threats causing vulnerabilities in the system. Abuse frames help us identify the scope of the threats affecting the system. It helps the developer in identifying which subproblems are vulnerable.

Step 4: Security requirements elicitation

In this step, we will model the security requirements that are intended to mitigate the threats causing vulnerabilities. We will model such security requirements using security problem frames [12] that are available in the security catalog. The security catalog will be used to retrieve the appropriate security problem frames corresponding to threats identified in step 3. Such security problem frames are generic, and thus need to be instantiated to the system context we are modeling. We will instantiate such security problem frames by modifying its domains, phenomena, interfaces, and security requirements. The output of such step will be *security problem frames diagrams* that model the security requirements. The requirements will be in the form of constraints on the functionality of the system focusing on *what* should be achieved.

For example, the corresponding security problem frame for the tampering threat in the catalog is "Integrity preserving of stored data." We will instantiate such security problem frame to model the security requirement that meets with the context of the system. The instantiated security problem frame (SPF) will be "SPF1: Integrity preserving of account information" shown in Fig. 5. In such figure, we have the *Authorized Bank staff* and *Unauthorized User* domains to represent the possible users of the system. The security requirement constrains crediting fund to accounts to be allowed only to authorized *Bank staff*.

In some cases, we might have difficulties in instantiating the security problem frame. For example, domains of the security



a: AI!{Content of account info.}
b: ABS!{Enter Identity , Commands to update content of account info.}
c: UU!{Enter Identity , Commands to update content of account info.}
d: IPM!{Update content of account info.}

SR: *Modification or creation of account information is allowed only to authorized bank staff and not allowed to unauthorized users*

Fig. 5 SPF1: Integrity preserving of account info.

problem frame may not match with the context of the system or the catalog may not include an appropriate security problem frame. In this case, we will follow the technique used in Haley's approach [13] to elicit the security requirements. We will update the problem frame diagram of the subproblem to consider security. We will *constrain* the requirements in order to mitigate the threats.

Trust assumptions might be used in order to mitigate discovered threats. Such assumptions state the analyst's beliefs that the properties of system domains can be trusted to an acceptable level that makes the system safe from vulnerabilities. By using a trust assumption, the analyst is putting bounds on the problem that the system must solve [13]. Trust assumptions are used only if the analyst is unable to go further through the problem because it is believed to be solved in another context [13]. For example, we can have *Authentication Data* domain that saves the credentials of the users. It can be given that such domain is under the responsibility of the IT department and that they are secure. These assurances can be given as trust assumptions.

Step 5: Security requirements evaluation

We will adopt a checklist from [22] in order to evaluate the resulting security requirements. Such criteria list the software assurance community concepts of goodness for security requirements. The criteria aim at providing good security requirements that are feasible, unambiguous and non-conflicting with other requirements.

We will also utilize a systematic approach inspired from [23] in order to identify conflicts between requirements. In order to check for conflicts between security requirements itself or between security requirements and functional requirements, we need to check how the common domains between the subproblems are constrained. We need to check whether there exist any conflicts in the constraints applied on such domains in the subproblems of the system. For example, we can have two conflicting requirements in a system for banking services. The first subproblem has a requirement that constrains the domain *Account info* to be edited by bank customers while another subproblem has a requirement that constrains the *Account info* domain to be prohibited from editing by bank customers. In this case, we have two subproblems that cause two conflicting constraints on the same domain. Any discovery of deficiencies in the requirements should return us to step 4 in order to refine the requirements and resolve the conflicts or ambiguity.

Methodology iterations and outputs

As shown in Fig. 1, the methodology iterates through its steps. Iteration starts from step 1 till reaching step 5. In each iteration, the subproblems frame diagrams may be updated to elaborate more domains in the system, and thus new assets in the system may be revealed that require new security requirements. For example, after eliciting security requirements in step 4, we can have new security problem frames, and thus new assets and threats may be elaborated. The iterations will stop when we do not find threats and vulnerabilities in the system.

The output of the approach is expected to be a group of (security) problem frame diagrams representing the subproblems that model functional or security requirements. Such requirements should ensure that the system to be developed is secure assuming that they will be satisfied in the further stages of software development. The output subproblems of the methodology will be an input to the composition stage in the problem frames based approach for software development. In our work, we are focusing on decomposing the system problem into subproblems while composing such subproblems together can be considered in future work.

Security catalog

Our proposed security catalog contains security models for *threats* and the corresponding *security requirements*. Abuse frames [11] are used to model threats while security problem frames [12] are used to model security requirements.

Such catalog is intended to be *generic*. It is not limited to specific domain context, and thus it can be customized and instantiated according to context of the software system to be modeled.

Catalog contents: Threats

The threats in the catalog will be modeled by abuse frames [11]. Abuse frames represent security threats that can be exploited by attackers or malicious users in specific problem context. Such threats will describe *what* the attacker can do in order to violate the security concerns of the system. We will utilize abuse frames because it has the advantage of bounding the scope of security problems. Thus, threat analysis can be performed on specific subproblems so that we can know what threats can affect which asset domains in which subproblems.

We introduce new abuse frames for commonly known threats. Such threats are classified by the categories of STRIDE [19] (spoofing, tampering, repudiation, information disclosure, denial of service and elevation of privilege). Such threat representation and classification can help the developer when using the catalog. We are not claiming to cover all possible threats that can affect software systems. However, STRIDE can assist us cover a wide range of threats that can violate the security concerns such as confidentiality, integrity, availability, accountability and authenticity.

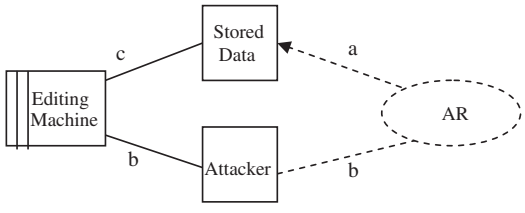
An example for a threat in the catalog is shown in Fig. 6 where the abuse frame **(AF)** "AF: Tampering stored data" is shown. A vulnerable *Editing machine* that is concerned with editing operations is being threatened by a tampering threat. The threat in Fig. 6 is a generic threat where the attacker modifies the stored data without authorization. The abuse frame used in representing such threat shows the interface between the system and the attacker. For example, the connecting line between the *Editing* machine and the *Attacker* **(ATT)** domain represents the interface having the notation **ATT!{Commands to edit Content of Stored data}**. Such notation denotes that the *Attacker* **(ATT)** domain is trying to make unauthorized editing to the *Stored Data* domain. The arrow headed dashed line that connects the anti-requirement **(AR)** and *Stored Data* domain represents a requirement reference having the notation **SD!{Content of stored info. after attack}**. Such notation denotes that the anti-requirement **(AR)** constrains the properties of the *Stored Data* **(SD)** domain after executing the attack. Such anti-requirement constrains the content of stored info to be modified by the *Attacker* domain without authorization.

Catalog contents: Security requirements

In this paper, we will adapt the definition of security requirements that represents positive statements representing constraints on the system behavior. We will adapt security problem frames [12] in order to model security requirements.

Security problem frames help model known security problems and represent the security requirements needed to mitigate threats. We will utilize some previously made security problem frames in addition to new ones that we introduce. For example, the security problem frame (SPF) "SPF: Integrity-preserving stored data" shown in Fig. 7 (inspired from [24]) represents a security requirement that is concerned with preserving the integrity of the stored information. Such security requirement mitigates the threat modeled by the abuse frame "AF: Tampering stored data" in Fig. 6 because the security requirement allows only authorized users to modify the stored data and prohibits unauthorized users from modifying it.

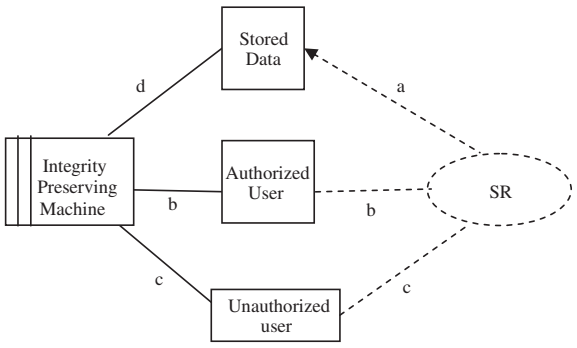
The interfaces in Fig. 7 show how the *Authorized user* and *Unauthorized user* domains interact with the *Integrity Preserving Machine*. For example, the connecting line between the *Integrity Preserving Machine* **(IPM)** and the *Unauthorized user* **(UU)** domain represents the interface having the notation **UU!{Enter identity, Commands to update content of stored data}**. Such notation denotes that the *Unauthorized user* domain performs



a: SD!{Content of stored info. after attack}
b: ATT!{Commands to edit Content of Stored data}
c: EM!{Update content of stored data}

AR: *Attacker makes modifications to stored data without authorization*

Fig. 6 AF: Tampering stored data.



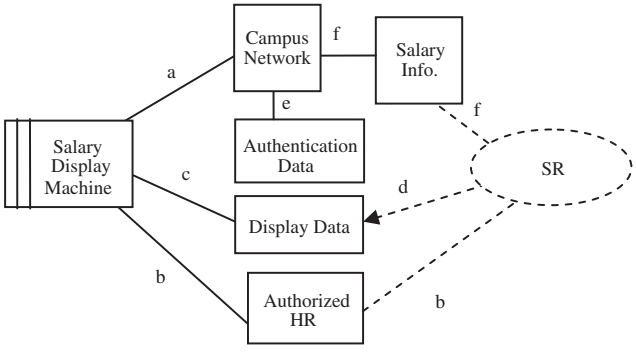
a: SD!{Content of stored data}
b: AU!{Enter identity , Commands to update content of stored data}
c: UU!{Enter identity , Commands to update content of stored data}
d: IPM!{Update content of stored data}

SR: *Modification or creation of stored data is allowed only to authorized users and not allowed to unauthorized users*

Fig. 7 SPF: Integrity preserving of stored data.

Table 1 Example of security catalog item.

Threat
Category: Tampering
Title: Tampering stored data
Abuse frame Fig. 6
Security requirement
Security concern: Integrity
Title: Integrity preserving of stored data
Security problem frame Fig. 7
Threat
Category: Tampering
Title: Tampering stored data
Abuse frame Fig. 6
Security requirement
Security concern: Integrity
Title: Integrity preserving of stored data
Security problem frame Fig. 7



a: SDM!{Read content of salary info, Send authentication data}
b: AH!{Enter identity, Request display}
c: SDM!{update content of display data}
d: DD!{Content of display data}
e: AD!{content of authentication data}
f: SI!{Content of salary info}

SR: *Authorized HR staff only is allowed to view salary information.*

Fig. 8 SPF2: Confidentiality preserving of salary information.

Table 2 Results of first iteration during applying our methodology on the HR system.

Step	Results
System modeling	Problem frames: "PF1: Salary Info editing" Requirement: *Salary info is edited by users* "PF2: Salary Info display" Requirement: *Salary information is displayed to users*
Identify assets	Assets: Salary Information
Identify threats and vulnerabilities	Abuse frames diagrams: "AF1: Information disclosure of salary information" AR: *Salary information is displayed to attackers without authorization* "AF2: Tampering Salary information" AR: *Attacker makes modifications to salary information without authorization*
Identify security requirements	"SPF1: Integrity preserving of salary information" SR: *Modification or creation of salary information is allowed only to authorized HR staff and not allowed to unauthorized users* "SPF2: Confidentiality preserving of salary information" SR: *Authorized HR staff only is allowed to view salary information*
Security requirements evaluation	The requirements complete each other and do not cause conflicts

Table 3 Results of second iteration during applying our methodology on the HR system.

Step	Results
System modeling	The security problem frame diagrams SPF1 and SPF2 will be modified by adding the domains *Campus Network* and *Authentication Data* to give more elaboration on the system as shown in Fig. 8
Identify assets	Assets: – Authentication Data
Identify threats and vulnerabilities	Abuse frames: "AF3: Information disclosure of transmitted salary information and authentication data" AR: *Salary information and authentication data are displayed to attackers while being transmitted* "AF4: Spoofing a HR staff" AR: *Attacker impersonates authorized HR staff and claims to be a valid accepted HR staff* AF5: Repudiation of salary info editing" AR: *Attacker makes changes to salary and denies performing it*
Identify security requirements	The security requirement in SPF1 will be as follows: SR: *Modification or creation of salary information is allowed only to authorized HR staff and not allowed to unauthorized users and the data sent over the campus network should not be understandable by eavesdroppers* The security requirement in SPF2 will be as follows: SR: *Authorized HR staff only is allowed to view salary information and the data sent over the campus network should not be understandable by eavesdroppers*
Security requirements evaluation	The requirements complete each other and do not cause conflicts

the operations of entering the identity and requesting to update the stored data. The security requirement **(SR)** references the *Authorized User* domain and the *Unauthorized user* domain in the requirement description. It constrains the contents of the *Stored Data* domain to be modified only by authorized users and not by unauthorized users.

Table 4 Results of third iteration during applying our methodology on the HR system.

Step	Results
System modeling	The domain *Campus Network* is updated to be *Encrypted Network* in an attempt to add a design solution to satisfy the security requirement that constrains the data sent over the campus network to be understandable by eavesdroppers.
Identify assets	Assets: No new assets
Identify threats and vulnerabilities	The threat described in AF3 is still applicable and can cause vulnerability because we are not sure if encryption keys are secure
Identify security requirements	The following trust assumptions are added: – *Encryption Network* domain uses secure encryption keys. – *Authentication Data* is secure
Security requirements evaluation	The requirements complete each other and do not cause conflicts

Security requirements will be categorized by security concerns. Such security concerns can be described as confidentiality, integrity, availability, accountability and authenticity.

Table 1 shows an example of a catalog item where each threat is linked with a corresponding security requirement.

Results and discussion

In this section, we compared our proposed methodology with two security requirement elicitation methodologies that are based on problem frames [12,13]. We have selected to compare with others [12,13] because our proposed methodology is inspired from such methodologies in the way of making use of problem frames in identifying vulnerabilities and eliciting security requirements in the system. We showed that the security requirements are more complete when following the systematic steps of our proposed methodology (see Fig. 8).

Comparing with Haley's methodology

We have applied our proposed methodology on the case study presented in [13] where Haley's approach is applied. Such case study is concerned with modeling security requirements for a Human Resources System. We have selected to compare with Haley's methodology because it is related to our methodology.

Table 5 The comparison results with Haley's methodology.

Haley's methodology results
Security requirements:
SR1: *only HR staff can edit or view salary information*
SR2: *information passing over the network must not be understandable by an eavesdropper*

The proposed methodology results
Security requirements:
SR1: *Authorized HR staff only is allowed to view salary information and the data sent over the campus network should not be understandable by eavesdroppers*
SR2: *Modification or creation of salary information is allowed only to authorized HR staff and not allowed to unauthorized users and the data sent over the campus network should not be understandable by eavesdroppers*
SR3: *Modification of salary information should be logged*
SR4: *The security validation state is accepted only if the user using identity of a HR staff is actually a HR staff*

We have applied the methodology according to the steps outlined in Section "Methodology" in order to model security requirements for the asset "Salary information" in the system. Tables 2–4 outline the results of each methodology iteration.

We have compared our results with those of Haley's methodology. The results showed that more threats are considered. This led to covering more security requirements. As shown in Table 5, the security requirement **SR3** is suggested to constrain the modification of salary information to be logged to preserve the accountability security concern. Furthermore, the security requirement **SR4** is proposed to ensure the validity of the HR staff identity during authentication to preserve the authenticity concern. Thus, more complete security requirements are elicited. We can argue that utilizing the security catalog enabled eliciting threats more systematically because of the assistance and suggestions provided by the catalog. The security catalog helped the elicitation of more effective security requirements that can mitigate and counter the threats because of considering more security concerns such as accountability and authenticity.

Comparing with security problem frames based approach

In this section, we will show the results of applying our proposed methodology on the case study presented in [12] where SEPP (Security Engineering Process using Patterns) is applied.

Table 6 The comparison results with SEPP methodology.

SEPP methodology results
Security requirements:
SR1: *Access is granted for the authentic user and access is denied for the Malicious subject*
SR2: *Malicious subject should not be able to derive sent data and received data during data transmission*
SR3: *Sent data equals Received data or if not, a modification by Malicious subject is detected using Transmitted data*

The proposed methodology results
Security requirements:
SR1: *The security validation state is accepted only if the user using identity is the one he claims to be*
SR2: *Transmitted screen data should be not understandable by attackers*
SR3: *Any modification to Transmitted screen data should be detected*
SR4: *Screen Data should be available for sending and displaying*
SR5: *Screen data changing and sending by users should be logged*

Such approach utilizes security problem frames to model security requirements. The case study on which SEPP is applied models security requirements for a Remote Display System. The case study presented in [12] is mainly concerned with modeling a secure Remote Display System. The system allows its users to view and control a computing desktop environment not only on the PC (Personal Computer) where it is running, but also from a PDA (Personal Digital Assistant) over a Bluetooth connection. Table 6 lists the results of applying such methodology in addition to the results of applying our methodology on the same case study in order to elaborate the advantages of the proposed methodology.

As shown in Table 6, we took into consideration more threats and this led to covering more security requirements. The security requirement **SR4** is proposed to ensure availability of the remote display service. Moreover, the security requirement **SR5** is suggested to log the remote display requests of the users in order to preserve the accountability security concern and mitigate the repudiation threat. Such security requirements are not covered when applying the SEPP approach. Thus, more complete security requirements are elicited when following our methodology because of considering more security concerns such as accountability.

Conclusions

We have suggested a methodology that integrates security with requirement engineering process based on problem frames. The proposed methodology is a result of our own contribution in addition to the integration of many useful approaches. We will list a summary of our contributions:

- We have developed a methodology that enables discovering threats and eliciting its countermeasure security requirements in a systematic way.
- We have proposed a problem frames based security catalog that combines threats with corresponding security requirements in order to assist the analyst elicit security requirements by reusing security knowledge.
- We have utilized abuse frames [11] in order to construct different generic threats under different categories that helped in constructing a generic model based security catalog. Such approach helped in bounding the scope of security problems.
- We have made use of security problem frames [12] to model known security problems and represent the security requirements needed to mitigate threats in the security catalog.
- We have adapted from Haley et al. [13] the way of utilizing problem frames in identifying vulnerabilities in the system by crosscutting threats with the system requirements.
- We have made use of evaluation criteria [22] to evaluate the resulting security requirements concentrating on conflicts identification between requirements.

We have compared our methodology with other relevant methodologies to demonstrate the benefits of using the methodology presented in this paper. First, we compared with Haley's approach, a case study which is applied to a human resource system [13]. Two more security requirements are elicited when applying our methodology. Such security requirements considered the accountability and authenticity security concerns. Second, we applied the methodology on the case study presented in Hatebur et al. [12] where security problem frames based approach is used to model security requirements for a secure Remote Display System. Two more security requirements are elicited when applying our methodology. Such security requirements considered the accountability and availability security concerns. Thus, we have shown that more complete security requirements can be elicited by such methodology in addition to the assistance offered to developers to elicit security requirements in a more systematic way.

Future work

More empirical studies on large scale software systems are needed to evaluate the methodology. We can apply the methodology on more case studies by wide range of software developers having different levels of security knowledge.

Moreover, we can adapt a formal framework into the methodology instead of its informal language dependency. Representing the requirements formally can help in achieving preciseness and automation.

Furthermore, adapting a risk analysis approach in the methodology can be beneficial. Such step can be used in order to evaluate the risk of the threats before mitigating it. Such approach will help us prioritize threats accurately and identify its severity in addition to identifying the best approach to mitigate such threats.

Finally, the security catalog used in the methodology can be extended to support more generic threats and security requirements. Covering more categories in addition to covering more domains can help expand the security catalog and make it more complete. Such extension can enhance the methodology and let it assist in capturing more complete security requirements.

Conflict of interest

The authors have declared no conflict of interest.

References

[1] Internet Storm Center Statistics. <https://isc.sans.edu/submissions.html>.
[2] Grand Research Challenges in Information Security & Assurance. CRA; November 2003, <http://www.cra.org/Activities/grand.challenges/security/home.html>.
[3] Sindre G, Firesmith DG, Opdahl AL. A reuse-based approach to determining security requirements. In: Proceedings of the 9th international workshop on requirements engineering: foundation for software quality (REFSQ'03); 2003.
[4] van Lamsweerde A. Elaborating security requirements by construction of intentional anti-models. In: Proceedings of the 26th Int'l Conf Software Eng (ICSE 04). IEEE CS Press; 2004.
[5] Liu L, Yu E, Mylopoulos J. Security and privacy requirements analysis within a social setting. In: Proceeding of RE'03; 2003. p. 151–61.
[6] Sindre G, Opdahl AL. Eliciting security requirements with misuse cases. Requirements Eng 2005;10(1):34–44.
[7] Haley CB, Moffett JD, Laney R, Nuseibeh B. Security requirements engineering: a framework for representation and analysis. IEEE Trans Software Eng 2008;34(1):133–53.

[8] Firesmith D. Security use cases. J Object Technol 2003;2(3): 53–64.
[9] Mouratidis H. Secure information systems engineering: a manifesto. Int J Electron Security Digital Forensics 2007;1(1): 27–41.
[10] Jackson M. Analyzing and structuring software development problems. Addison-Wesley; 2001.
[11] Lin L, Nuseibeh B, Ince D, Jackson M, Moffett J. Introducing abuse frames for analyzing security requirements. In: Proceedings of the 11th IEEE international requirements engineering conference (RE'03), Monterey, CA, USA; 2003. p. 371–2.
[12] Hatebur D, Heisel M, Schmidt H. A pattern system for security requirements engineering. In: Proceedings of the international conference on availability, reliability and security (AReS), IEEE; 2007. p. 356–65.
[13] Haley CB, Laney R, Nuseibeh B. Deriving security requirements from crosscutting threat descriptions. In: Proceedings of the third international conference on aspect-oriented software development, Lancaster, UK; 2004.
[14] Whittle J, Wijesekera D, Hartong M. Executable misuse cases for modeling security concerns. In: Proceedings of the 30th international conference on software engineering, Leipzig, Germany; 2008.
[15] Giorgini P, Massacci F, Mylopoulous J, Zannone N. Requirements engineering meets trust management: model, methodology, and reasoning. In: Proceeding of iTrust-04, LNCS 2995. Heidelberg: Springer-Verlag; 2004. p. 176–90.
[16] Oladimeji E, Supakkul S, Chung L. Security threat modeling: a goal-oriented approach. In: Proceedings of SEA'06, Dallas, TX; 2006.
[17] Yin B, Jin Z. Extending the problem frames approach for capturing non-functional requirements. In: Proceedings of the 11th international conference on computer and information science; 2012.
[18] Schneier Bruce. Attack trees. Dr. Dobb's J; 1999.
[19] Swiderski F, Snyder W. Threat modeling. Microsoft Press; 2004.
[20] Hermoye LA, van Lamsweerde A, Perry DE. A reuse-based approach to security requirements engineering; 2006. <http://users.ece.utexas.edu/~perry/work/papers/060908-LH-threats.pdf>.
[21] Cote I, Hatebur D, Heisel M, Schmidt H, Wentzlaff I. A systematic account of problem frames. In: Proceedings of the 12th European conference on pattern languages of programs (EuroPLoP 2007); 2007.
[22] Information Assurance Technology Analysis Center (IATAC)/ Data and Analysis Center for Software "Software Security Assurance. A state-of-the-art report; 2007.
[23] Jackson M. The problem frames approach to software engineering. In: 14th Asia-Pacific, software engineering conference (APSEC'07); 2007. apsec, p. 14.
[24] Heisel M, Hatebur D. Security problem frames. Entwicklung Sicherer Software SS; 2007.