A Systematic Approach to Generate the Security Requirements

For the Smart Home System

by

Rongcao Xu

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Computing Studies

Approved May 2013 by the
Graduate Supervisory Committee:

Arbi Ghazarian
Ajay Bansal
Timothy Lindquist

ARIZONA STATE UNIVERSITY

August 2013

ABSTRACT

Smart home system (SHS) is a kind of information system aiming at realizing home automation. The SHS can connect with almost any kind of electronic/electric device used in a home so that they can be controlled and monitored centrally.

Today's technology also allows the home owners to control and monitor the SHS installed in their homes remotely. This is typically realized by giving the SHS network access ability.

Although the SHS's network access ability brings a lot of conveniences to the home owners, it also makes the SHS facing more security threats than ever before. As a result, when designing a SHS, the security threats it might face should be given careful considerations.

System security threats can be solved properly by understanding them and knowing the parts in the system that should be protected against them first. This leads to the idea of solving the security threats a SHS might face from the requirements engineering level.

Following this idea, this paper proposes a systematic approach to generate the security requirements specifications for the SHS. It can be viewed as the first step toward the complete SHS security requirements engineering process.

DEDICATION

This thesis is dedicated to my family for their unconditional love.

ACKNOWLEDGEMENTS

I wish to thank my supervisor, Dr. Arbi Ghazarian, for his great support and patience in this project and my learning and understanding the field of requirements engineering.

TABLE OF CONTENTS

LIST OF FIGURES

Chapter 1

INTRODUCTION

One of the major causes of the information system design failure is that its functionalities and performance do not meet its stakeholders' expectations. Problems in the requirements engineering process, such as the ambiguous and inconsistent requirements specifications, the inadequate verification of the specified system requirements and the improper responses to the requirements changing requests, are among the main reasons for causing such a failure (Sommerville, Software engineering, 9th edition, 2011) .

At the same time, most of the information systems today face more security threats than ever before as a result of their enhanced network access ability. Although this ability brings convenience to personal life, it also increases the possibility for the information systems to be attacked maliciously. If an information system is designed without the consideration of the security threats it might face then it is less useful or worthless.

Smart home system (SHS) is a kind of information system aiming at realizing home automation (Home automation). Its build in network access ability allows its users to know the status of their homes remotely and also allows them to control the facilities in their homes remotely.

The users of the SHS can also use it to do a lot of other useful things like the home entertainments, on-line chatting and the elderly monitoring.

Clearly, if there is no security mechanisms build into the SHS then it can be intruded by the hackers easily and the results are the hackers can control every home facilities connected with it, modify the critical data stored in it and report fake information about the home status to the home owners.

The discussion above indicates that the requirements engineering and the security mechanisms are among the key factors for a SHS to gain the commercial success in the market. As a result, this paper proposes an approach on generating the security requirements specifications of a SHS. The aim of this approach is to understand and solve the security threats a SHS might face from the requirements engineering level and to produce its security requirements specifications in a systematic way.

Following the introduction this paper is organized as follows: Section 2 reviews current the security engineering practices in the field and in the SHS; this section also identifies the problem to be solved by this paper. Section 3 proposes a systematic approach to generate the security requirements specifications for a SHS. Section 4 is the case study and section 5 summarizes the work done in this paper and points out the future directions.

Chapter 2

BACKGROUND AND RELATED WORK

Requirements engineering and security, as discussed in chapter 1, are key factors for a SHS system to gain commercial success in market and they are the backgrounds for further discussion. As a result this chapter gives some discussion on these factors.

2.1 View requirements as conditions to meet to solve problems

A system is useless if it does not do what the stakeholders need it to do; this is also true for the information systems.

The needs from stakeholders define the system at the most abstract level; because these needs just state what is needed and say nothing on how to design and implement such a system. As an example, a need from stakeholders could be "We need a new server system that can handle 100 transactions concurrently."

The needs from stakeholders also present the highest level problems that should be solved during system designing process. If these problems are properly identified and solved then the needs from stakeholders would be met. Stated in another way, meeting stakeholders' needs is the process of solving the problems presented in their needs.

Problems can be viewed as obstacles on the way of reaching desired results. Solving problems is the process of knowing the problems and finding conditions that can lead to the problems solutions.

Generally, solving problems is a multi-stage activity. For a given problem and in the first stage of problem solving, efforts are exerted to find conditions (ideally necessary and sufficient) that can lead to the problem solutions. But meeting these conditions may not be an easy task; as a result, these conditions become problems themselves and this leads us to the second stage of problem solving process. In the second stage, efforts are exerted again to find conditions leading to the problem solutions. This process is repeated until, at a certain stage, the conditions that lead to problem solutions can be met easily. These conditions are at the bottom of the problem solving process. The meeting of conditions at bottom solves the problems at the bottom stage; and, in general, the solved problems at a certain stage will solve the problems at the stage that is above it. Such a process is the general problem solving process.

Solving the problems presented in stakeholders' needs is similar to the general problem solving process. The problems should be identified as clear as possible and then conditions that can lead to the solutions should be identified. These conditions are in fact requirements for system; they are what we the system designers are required to do to solve the problem.

Also similar to the general problem solving process, these conditions may become the new problems the system developers should solve in system developing process.

Holding this view of point, requirements can be defined as a description of conditions that should be met by system designing and implementation at

successive system development stages. These conditions can be system's operational, functional characteristics or constraints.

2.2 The role of requirements in system design and implementation

Requirements play important role in system design and implementation. Without stable requirements the project is to be flounder (Elizabeth Hull, 2011).

With requirements, the system developers can evaluate if a design can fulfill the requirements or not and at what cost; can predicate what could be if the design fail to meet the requirements. When a changing request for a requirement raises, either from customer or development team side, its impact on the whole system can be evaluated; as a result one can determine whether to accept such a change or not.

In summary, the requirements are used for: project planning, risk management, acceptance testing, trade off and change control (Sommerville, Software engineering, 9th edition, 2011) (Elizabeth Hull, 2011).

2.3 Requirements engineering

Because system requirements are so important, a systematic way is needed to create, verify/validate and manage them. This is the place where the requirements engineering come in.

Requirements engineering (REE) as defined by Elizabeth Hull et al is "*the subset of systems engineering concerned with discovering, developing, tracing,*

analyzing, qualifying, communicating and managing requirements that define the system at successive levels of abstraction." (Elizabeth Hull, 2011).

Sommerville has defined RE as *"The process of finding out, analyzing, documenting, and checking the services and constraints is called Requirements Engineering (RE)"* (Sommerville, Software engineering, 2007).

As a result, the main steps in REE process are requirements elicitation, analysis/derivation, specification, verification/validation and management.

2.3.1 Requirements elicitation

Requirements elicitation is the process to identify system requirements from stakeholders' needs. They are the system requirements that define the system at the highest abstraction level.

2.3.2 Requirements analysis/derivation

Requirements analysis/derivation means to analyze system requirements which define system at certain abstraction level and derive new system requirements from them.

2.3.3 Requirements specification

Initially, requirements elicited from stakeholders' needs or derived from system requirements that define system at a certain abreaction level would be an (vague) idea in the requirements engineers' mind. They should be expressed out in some way so that they can be analyzed and inspected concretely. The process to express requirements concretely is called requirements specification.

Depending on the actual need, the requirements can be specified informally by using natural language or formally by using formal language.

Requirements specifications should be clear (does not contain ambiguity), consistent (does not contradict with each other), complete (enough for developing solutions) and testable (be used as a testing acceptance criteria) (Elizabeth Hull, 2011). But most importantly, they should reflect the “real” meaning of system requirements.

2.3.4 Requirements verification/validation

Requirements verification/validation is the process of checking that the specified requirements actually define the system that the customer really wants (Sommerville, Software engineering, 9th edition, 2011). If they are not, the requirements would be re-analyzed and re-specified.

As a result the requirements analysis, specification and verification stages are not totally separated, they are related with each other.

2.3.5 Requirements management

Requirements management is to manage the requirements after they are specified and verified. Activities in requirements management include requirements changing and tracing management.

2.3.6 Requirements engineering process at successive system developing stages

A system development process is carried out at successive system abstraction levels, starting from the most abstract one and ending at the most concrete one. These successive system abstraction levels are corresponding to successive

system developing stages. Each of the stages would impose some requirements to its successive stage. As a result, ideally, for each of the system developing stages, the requirements engineering cycle would be repeated. This can be summarized in the following Figure 1 (Elizabeth Hull, 2011)

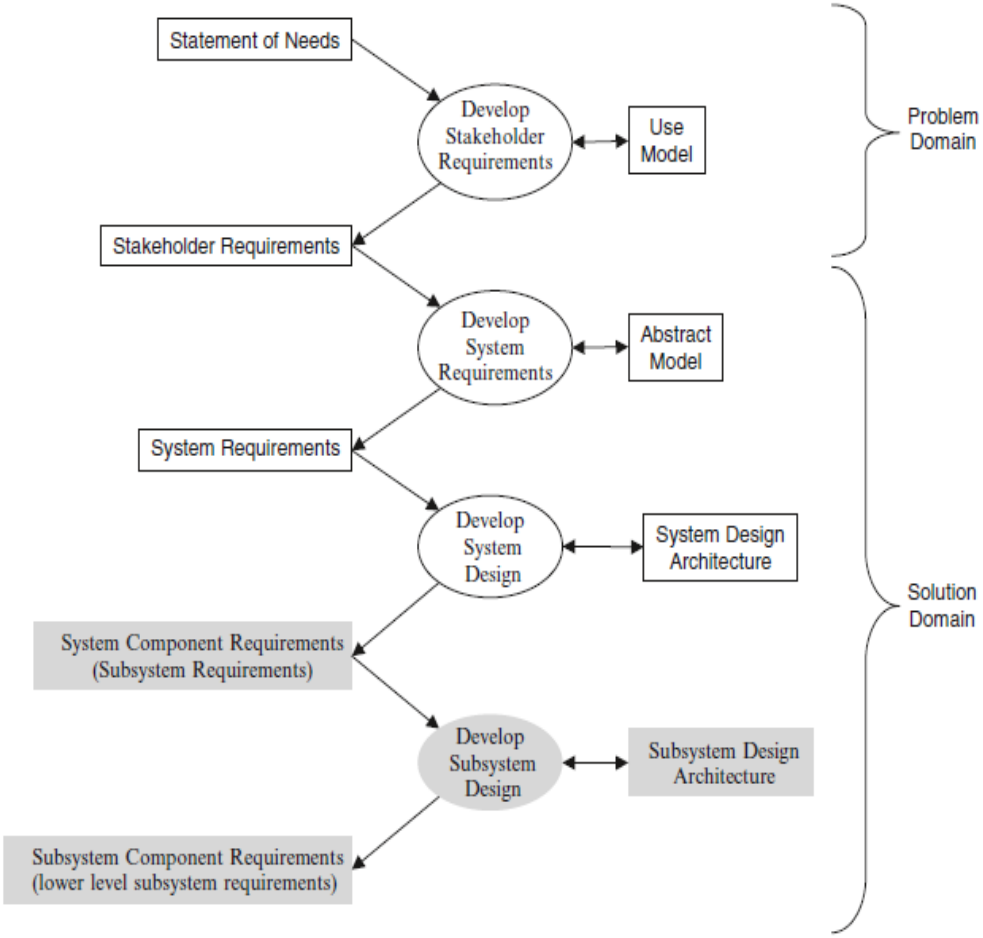

Figure 1: Requirement engineering process

The problem domain is the domain in which a system is going to be used to solve the problems (needs) presented by stakeholders (Elizabeth Hull, 2011). The

requirements engineering process in problem domain aims at forming stakeholder requirements that define the system at highest abstraction level.

The solution domain is where engineers use their ingenuity to solve problems (Elizabeth Hull, 2011). Requirement engineering process in the solution domain is to derive system requirements at successive system abstraction levels.

If the system developers, when developing the system, jump directly into the solution domain, the chances are the functionalities the system provides are not those expected from system stakeholders.

2.4 Security and sound security engineering practice

Hardware, programs and data are system assets in an information system and malicious persons can bring harms to them. For example, system CPU and hard disk can be physically destroyed; confidential data can be read, copied, deleted or altered maliciously. The consequence of allowing the harms to happen is typically a disaster (Philip Koopman, Carnegie Mellon University, 2004), (Charles B. Haley, 2008).

Security in the information systems is the mechanism to prevent such harms from happening. Harms can be viewed as security threats and system assets that are under the threatening of security threats are security assets.

Because of the booming of mobile applications, the wide availability of network access and the increasing system interconnection, security problems in the information systems today are more severe than ever before. If there is no

security mechanism being designed into the information systems, malicious persons can bring security threats to them easily.

Security is easier to archive if there is a clear model of what is to be protected and who is allowed to do what (Tanenbaum, 2007). Some security engineering practitioners also suggest that system developers should firstly establish a sound security policy and treat security as an integral part of system design to solve security problems. (Rushby, 2001), (Charles B. Haley, 2008), (Guttorm Sindre, 2004)

This makes sense that a sound security engineering practice should start from requirements engineering level. Beginning with stakeholders' security needs, system developers proceed to generate stakeholders' security requirements; then the system security requirements are derived from them and finally security solutions are developed according to the system security requirements.

A finished system could be either over protected or under protected if security problems are solved directly from solution domain. Over protecting means additional cost, under protecting means there are security leaks in system. Both of them are not expected.

2.5 Smart home system (SHS) and its security problem

Smart home system (SHS) is a kind of information system aiming at realizing home automation. Anything in a home that uses electricity can be put on the SHS and at your command (Goodwin, 2010).

A SHS can be viewed as a collection of nodes and each node has its dedicated functionalities. Some of the nodes in the SHS are sophisticated embedded systems. They have powerful CPU and operating system; they run application code in RAM instead of ROM and they store files in flash storage system. Stated in another way, they are more likely a traditional PC system; for example, there are SHS running Linux seen in market (Goodwin, 2010).

The PC like feature and the build-in network access ability in SHS make them also facing the security threats seen in desktop and server system. Clearly, when a SHS is developed security problems must be considered.

Chapter 3

PROBLEM STATEMENT

A key feature for information system is security; but how are security problems understood and solved in practice and particularly in SHS?

3.1 Security engineering practice as an add on feature

"*Information systems security issues have usually been considered only after the system has been developed completely, and rarely during its design, coding, testing or deployment process*" (Ambrosio Toval).

"The security is often treated by embedded system designers as the addition of features to the system" (Paul Kocher, 2004).

These statements indicate that the security is not considered as an integrated part of the system design process. Instead, it is often considered when the designing of other system features is finished.

One of the consequences of solving the security threats a system might face in this manner is the system's structure could be destroyed by the added security mechanisms; another consequence is that the system performance can be severely affected by the added security mechanisms.

As a result, many security engineering practitioners argue to understand and solve the security threats a system might face from the requirements engineering level.

3.2 Security engineering practice as a part of requirements engineering process

What is the situation if the security threats are understood and solved from the requirements engineering level?

John Wilander et al present a survey on the security requirements engineering in their paper (John Wilander and Jens Gustavsson, 2005). The paper surveyed the security requirements specifications in 11 different kinds of information systems. They are the billing, accounting, e-business and other information systems.

The outcome of this survey is not optimistic. One conclusion presented in the paper is: *"the security requirement is poorly specified."* The main problems in the specification are the inconsistent selection of the security requirements and the inconsistent level of details of the specified security requirements.

Inconsistent selection of the security requirements means some relevant security areas are fairly well specified whereas other are completely left out (John Wilander and Jens Gustavsson, 2005).

As examples, John Wilander et al point out in their paper that all systems surveyed have requirements which indicate that restricted access is important. At the same time only three specifications require some kind of encryption of data communication and only two specifications require physical security including restricted physical access.

Inconsistent level of detail means some security requirements have a high level of detail whereas others in the same specification are only specified on a general level (John Wilander and Jens Gustavsson, 2005).

John Wilander et al also give an example on this factor: "*This phenomenon can be seen in for instance the E-Business system where the requirements on logging are very detailed (eight requirements on what info to be logged) and at the same time digital signatures are specified as: The system should be able to handle the use of electronic signatures with no further details.*"

This conclusion indicates that there are still problems in the security engineering process even if the security issues are understood and solved from the requirements engineering level.

Another important conclusion presented in this survey is that most of the security requirements are functional instead of being non-functional. Because the survey result reveals more than 75% of the security requirements (164 out of 216) boils down to functional requirements. This is a good indication that most of the security requirements can be formalized.

3.3 Security engineering practice in SHS

Papers, like (Rosslin John Robles, 2010), (Eržen, 2012) (Mohd Ariff Razaly, 2012), (Georgios Mantas, 2011) and others, are trying to discuss the countermeasures to address a particular or some kinds of security threats a SHS might face.

Although these papers provide great information, by just putting the countermeasures proposed in those papers blindly into the SHS under development may result in system overprotection or under protection. This may also bring the unnecessary development cost.

In another words, in order to build proper and efficient security mechanisms for a particular SHS under development, there is a need to know exactly what are the security threats it might face first; then the system developers can define security requirements for it and finally proper countermeasures proposed in these papers can be selected or new countermeasures can be established to deal with those security threats.

This means the security problems in the SHS should be understood and solved from the requirements engineering level. But none of these and others papers, discussing the security threats in the SHS, surveyed so far addresses this issue.

Taking this discovery and the poorly specified system security requirements as problems, this paper proposes an approach to systematically generate the security requirements specification for a SHS under development. The generated specifications would be consistency in both of the selection of security requirements and the level of details. This approach can be viewed as the first step toward the complete SHS security requirements engineering process.

Chapter 4

PROPOSED SOLUTION

4.1 Introduction:

The solution is to understand and solve the security problems in the SHS from requirements level. The focus of this solution is on requirements specifications; because (sound) requirements specifications are the basis for further system development.

4.2 Solution process:

Three main steps are needed to generate the SHS security requirements specifications and they are depicted in Figure 2.

Step 1 analyzes the security needs proposed by stakeholders. These needs define the system's security features at the highest level and they are abstract. Nothing is mentioned on what parts in the SHS should be protected and how to protect them. So one of the main purposes of this step is to identify the security threats a SHS might face and the parts in it that should be protected.

The output of this step is a SHS security threats profile and it contains five parts. They are, in turn, the abstract system model; the system node information table; the security threats check list; the security assets/threats table and the abuse cases. The security threats profile is the basis for defining system security requirements.

Step 2 defines the high level security requirements for each of the parts in the SHS that should be protected by some kinds of security mechanisms. They are

described abstractly by just stating that a system part should be protected from a particular security threat. The threats profile generated in step 1 is used to assist this step.

Step 3 derives the functional system security requirements from the high level abstract security requirements defined in step 2. These derived functional security requirements are specified informally in natural language. This step is also assisted by the threats profile.

Step 1 can enhance the consistency in the selection of the security requirements because it forces the system developers to identify the security threats a SHS might face as completed as possible and as a result there is less chances to leave out security requirements on some security relevant areas.

Step 2 and 3 can enhance the consistency in the level of specification details because they require the system developers to define the high level security requirements first and then to derive the functional security requirements.

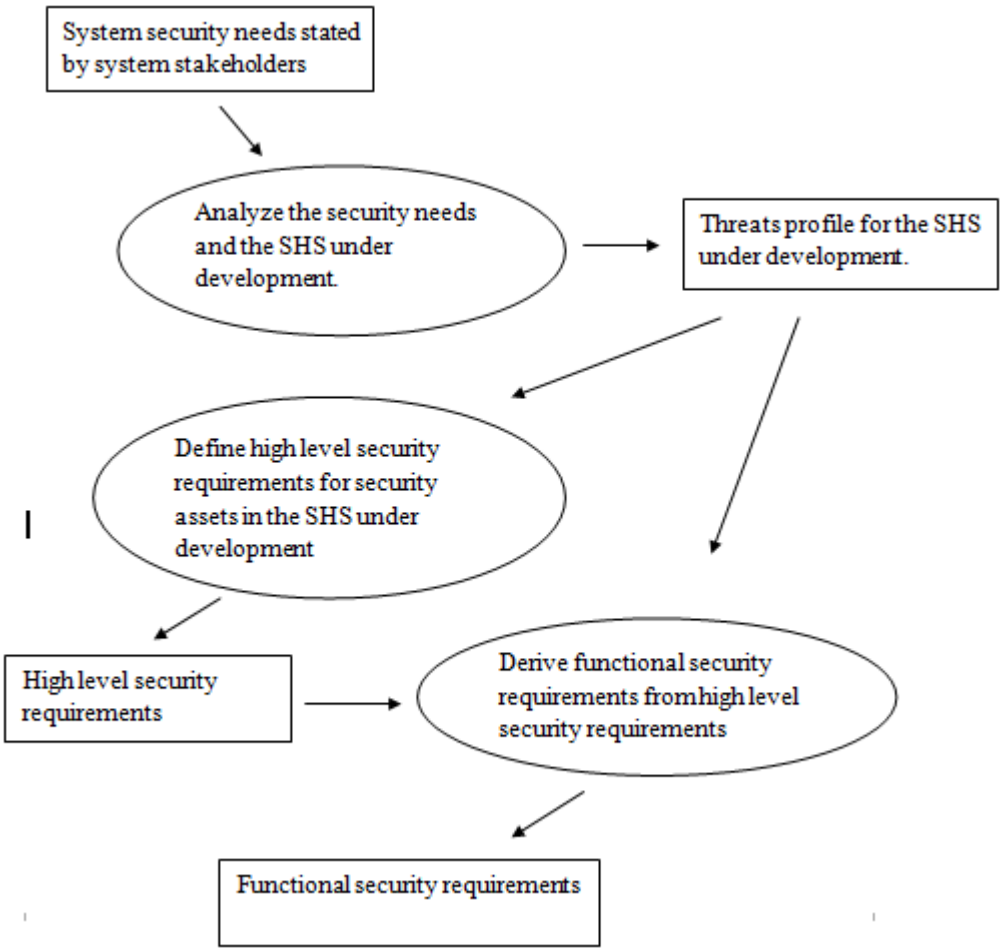

Figure 2: Three steps to generate the SHS security requirements

4.3 Detailed discussion

As discussed in chapter 2, system assets in an information system are system hardware, software and data/information stored in such a system. The security assets are those system assets that face security threats.

4.3.1 Creating the SHS security threats profile

The first step towards the sound security requirements engineering process in any information system is to know what security threats it might face and the parts that should be protected from them. Implementing security mechanisms

without knowing the security threats may result in waste of money and effort (Suvda Myagmar), (Dale R. Thompson) (Microsoft, J.D. Meier, Alex Mackman), (Tanenbaum, 2007).

A systematic approach is needed to create the SHS security threats profile; creating it without carefully planned steps would also result in improper security threats solutions. Figure 3 depicts the steps used to create the SHS security threats profile.

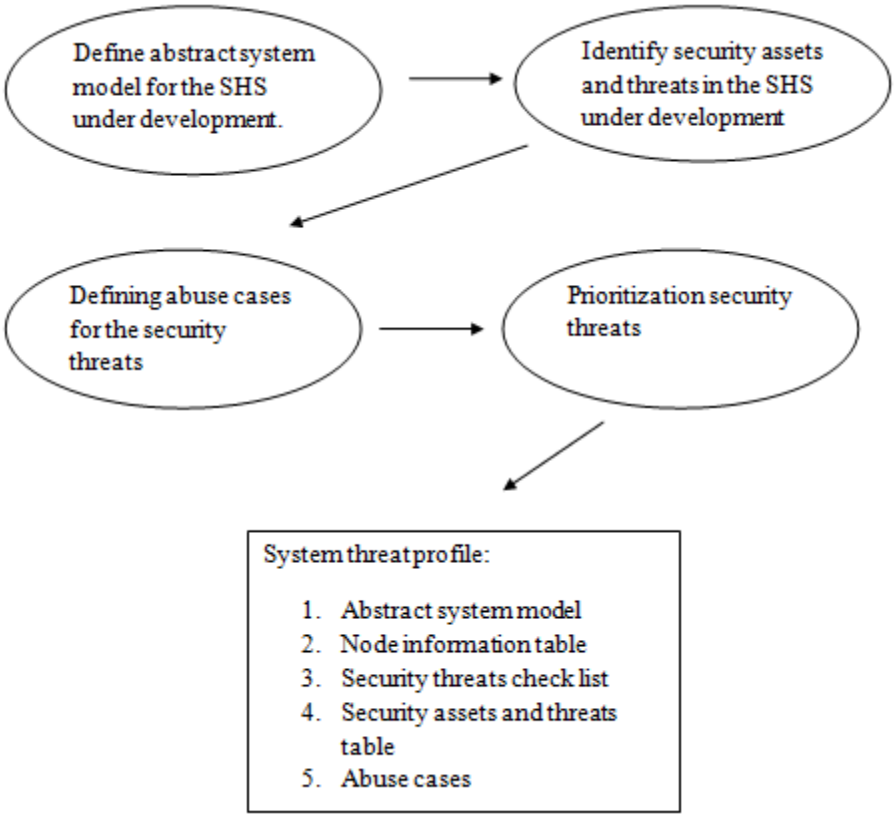

Figure 3: Steps to create the SHS security threats profile

4.3.1.1 Define abstract system model for the SHS under development:

The SHS is not a new conception and there are a lot of commercial SHS on the market. Based on the researches made on these SHS, a general SHS abstract

model is defined and is depicted in Figure 4. (This is not to say the proposed solution excludes other general abstract model)

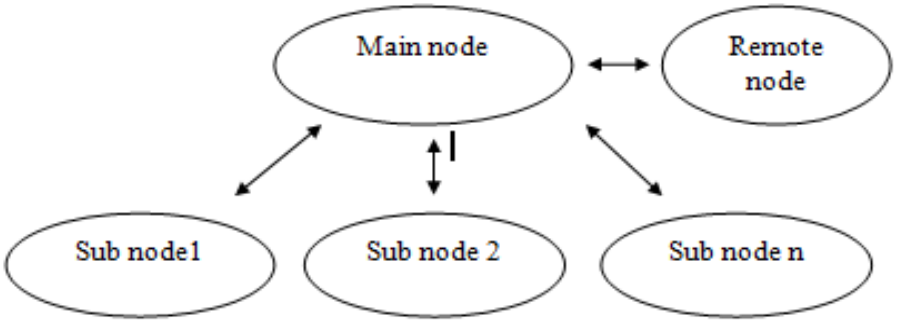

Figure 4: A general abstract SHS model

In essence, a SHS is a kind of communicating network composed by different system modules and communication links. Each module has its dedicated functionalities. For example the main control module is responsible for receiving user commands, coordinating the operation of the system and reporting the system status to the home owners. And all the modules in the SHS are working together to provide the overall SHS functionalities.

The ovals in the abstract system model represent the different system nodes in a SHS. A single node is an abstraction of a particular module in the SHS. For example, the main node abstracts the main control module and a security sub node abstracts the security module in the SHS.

The double heads arrow is the abstraction of the communication links between the system modules. In reality these links could be in either wired or wireless

form. The double heads arrow also indicates that the link is usually a two way communication link.

It is possible for a node to be as complicated as a communicating network itself. But these details are ignored and the focus is on the overall functionalities it provides.

It is common for the SHS to be accessed through a PC, a smart phone or a tablet. The specially designed SHS control software is running on these devices. The users can use this software to control and monitor the SHS remotely. Based on this observation, for a SHS, these kinds of devices are abstracted as a remote node.

If a SHS under development has the structure described in this general abstract model then its abstract system model can be an instance of it.

The main purpose of this step is to reduce the number of system assets to be evaluated against security threats. When doing the evaluation, instead of evaluating every system assets, only the abstract system nodes and the communication links are considered.

4.3.1.2 Identify security assets and security threats in the SHS

This stage identifies the security assets in a SHS and the security threats they may face. Figure 5 describes the steps to identify security assets and security threats in a SHS.

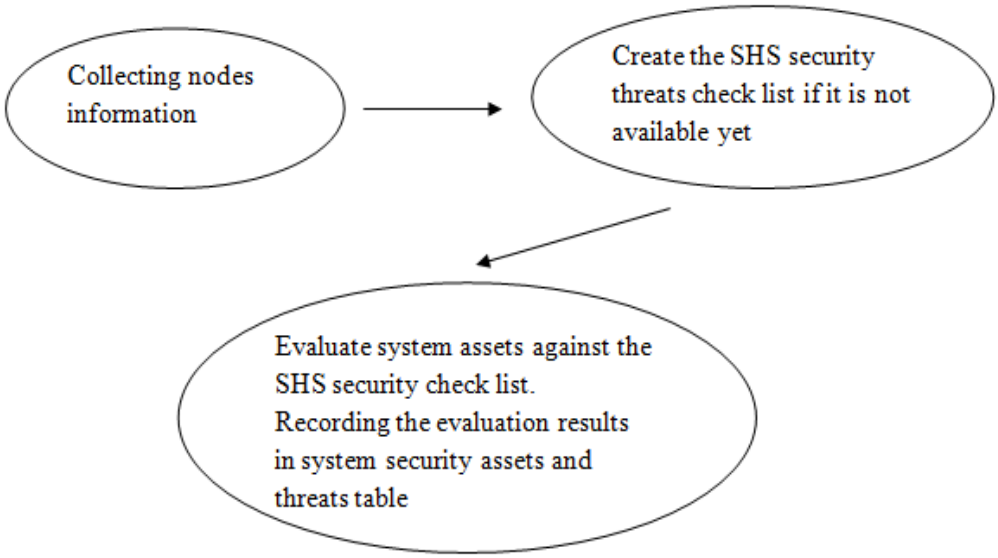

Figure 5: Steps to identify security assets and security threats in SHS

In the first step, related information is collected for the nodes identified in the abstract SHS model. The information to be collected is the functionalities, interfaces, communication links and data of a node. Additional information may also be collected depending on the actual needs.

The collected information could be recorded in the nodes information table. The table is depicted in Figure 6. Typically, such information could be obtained by reading other system development documents and the collected information is used to assist the determination of the security threats a node might face.

Node information	Node 1	Node 2	Node3	….Node n
User Interface				
Functionalities				
Communication links				
Data on node				
Others				

Figure 6: Nodes information Table

The second step, if the security threats checklist is not available yet, is the creation of the security threats checklist for the SHS.

The checklist is used to list possible security threats the SHS might face and once it is created the system developer can use it to evaluate whether the system is under the risk of a particular security threat. The checklist can also be updated as needed and reused when developing similar systems in future.

A SHS might face many security threats and sometimes they seem to be complicated but some of them may fall into a particular category. As a result security threats categorization could be an effective way to manage the complexity.

The main parts in the SHS are network interface, nodes and internal communication links and they are the targets of the security threats. As a result, the security threats a SHS might face can be categorized by considering whether they are targeting at the network interface, the nodes or the internal communication links. Based on this observation the security threats checklist for the SHS is defined and it is shown in Figure 7.

Threats category	Threat name	Description
Network interface		
Node		
Internal communication link		

Figure 7: Security threats checklist table

The system developers can focus on the network interface first when building the security threats checklist. The possible security threats on the network interface are recorded into the table with their names and descriptions. The same process is applied to the internal communication links and nodes.

When the nodes information table and the security threats checklist are ready, the third step is to evaluate the system nodes in the SHS under development against the security threats checklist. The user interfaces, functionalities, communication link and data on a particular node is evaluated in turn to see whether they are under the risk of the security threats listed in the checklist. If the answer is yes then a node is determined as a security asset. The evaluating results are recorded in the SHS security assets and threats table. This table is depicted in Figure 8.

Threat Category	Threat name	Node 1	Priority	Node n	Priority
Network interface	Spoofing	Y	2.6		
	Eavesdropping	Y	2.5		
Internal communication link					
Node					

Figure 8: SHS security assets and security threats table

If the node is under the risk of a specific threat then the corresponding unit in the table is marked as “Y”.

The priority of a particular security threat imposed on a node is also recorded in this table. But this information is obtained and filled in the prioritization step.

4.3.1.3 Defining abuse case for security threats and determining their possibility and risk level

An abuse case for each of the security threats a node might face is defined in this stage. Abuse case is the descriptions of the interactions between a hacker and a system that carry out the security threats on system which resulting in harm (John McDermott).

The purpose of defining abuse case is to force the system developers to think as a hacker so that a better understanding on how a particular security threat can be carried out is obtained. Such an understanding is helpful in determining the possibility and the risk level of a particular security threat.

This understanding is also helpful in knowing what methods can be used to prevent these steps from being carried out.

The abuse case would be recorded in the template depicted in Figure 9.

Abuse case number				
Abuse case name				
Source	Node	Threat category	Threat name	
Purpose				
Steps				
1				
2				
3				
… n				
Consequence				
Possibility criteria	Skill Level	Time Cost	Money Cost	Average
Score				
Possibility				
Risk level criteria	Damage potential	Reproducibility	Number of user affects	Average
Score				
Risk level				

Figure 9: Abuse case template

In the template, the abuse case number is the reference number for a particular abuse case; the abuse case name gives a description of what it is about; the source section records the originating of this abuse case; the purpose section illustrates the goal of carrying out this particular security threat; the steps section summarizes the main steps used to accomplish the goal; the consequence section records the possible results if such a security threat is carried out successful.

The possibility section records the chance for this security threat to happen; the risk level sections records the severity of the consequence and these two factors are used to determine the priority of a threat.

Possibility is quantified against three criteria: the level of skills needed, time and money cost needed to perform the attacks. For a particular security threat,

these criteria are scored within the range of 1 – 3. The scores are recorded in the corresponding locations in the abuse case template; they are summed and averaged. The possibility is the reciprocal of the averaged result, so the possibility value ranges from 0.3-1. Under this algorithm, the higher the average is the less possibility for it to occur. Inspired by, (Microsoft, J.D. Meier, Alex Mackman) (Suvda Myagmar), (EbenezerA.Oladimeji), Figure 10 is the table that records the possibility scoring criteria.

Criteria	Level and Score		
	High, 3	Medium, 2	Low, 1
Skill level	Expert level knowledge on system architecture, system communication protocols and system working principles is needed.	Medium level knowledge on system architecture, system communication protocols and system working principles is needed.	Entry level knowledge on system architecture, system communication protocols and system working principles is needed.
Time cost	Over 8 weeks	Within 4-8weeks	Less than 4 weeks
Money cost	Expensive devices and software are needed	Medium price devices and software are needed	Cheap devices and widely available software are needed

Figure 10: Possibility scoring criteria

Risk level quantification is also done by scoring a set of evaluating criteria in the range of 1 – 3 then summing these scores and averaging the sum. Damage potential, reproducibility, and number of users affected are the criteria for evaluating the risk level. The scores are also recorded in the abuse case template.

It is intuitive that the higher these scores are the higher risk level a security threat can impose on the system. Figure 11 is the table records risk level scoring criteria (Microsoft, J.D. Meier, Alex Mackman).

Criteria	Level and Score		
	High, 3	Medium, 2	Low, 1
Damage potential	The attacker can subvert the security system; get full trust authorization; run as administrator; upload content	Leaking sensitive information	Leaking trivial information
Reproducibility	The attack can be reproduced every time and does not require a timing window	The attack can be reproduced, but only with a time window and a particular race situation	The attack is very difficult to reproduce, even with the knowledge if the security hole
Number of users affected	All the users	Part of the users	Only one or two users

Figure 11: Risk level scoring criteria

4.3.1.4 Prioritizing threats

Due to the cost, time to market and other constraints, it is impossible to mitigate all the possible security threats a system might face (Suvda Myagmar).

At the same time not all the security threats would have the same importance level. Some of them would bring severe consequence to the system and is very likely to happen; some of them are severe but are unlikely to happen; while others are less likely to happen and would bring inconsequence harm to the system. So there is a need to evaluate the priority of the security threats. By knowing the priority, the system developers may exert their efforts to solve high priority security threats and they may decide to accept the happening of a low priority security threat.

Priority should be quantified. The priority number would help the system developers to understand the severity of a particular security threat.

A simple algorithm is used in this approach: Priority = Possibility * Risk level (Suvda Myagmar), (Microsoft, J.D. Meier, Alex Mackman). Possibility and risk level are already determined when defining abuse cases so that the calculation of priority is straightforward. The calculated priority values are recorded into the SHS security assets and threats table.

If the priority of a security threat imposed on a node is lower enough then the system developers may chose to accept such a threat to happen.

After these 4 steps, the SHS security threats profile is generated. It has five parts: an abstract system model, nodes information table, a security threat check list, a system security assets and threats table and a collection of abuse cases. This profile would give the system developers a good understanding of the threats the SHS under development might face; how these threats can be carried out and their priorities. This profile is also the basis for defining system security requirements.

4.3.2 Defining high level security requirements

The system developers would use the SHS security assets and security threats table to assist this defining work.

For each of the nodes in the table, if it is under the risk of a particular security threat and if the threat has a high priority value then a high level security requirement is defined on it; if the threat has a low priority value then the system developers may chose to allow the threat to happen and there is no high level security requirement defined against that threat.

For example, if the main control node in a SHS is under the threat of spoofing security threat and the priority value is 3 then a high level security requirement can be defined as: “The main control node should be protected from spoofing threat whenever there is an attempt to access it from network.”

The defined high level security requirements are recorded in the security requirement table which is shown in Figure 12.

The category section indicates this requirement is security requirement; the number section contains the numerical identification of this security requirement; the name section describes the name of the security requirement; the source section is used to identify the originality of this security requirement.

The derived functional security requirements are also recorded in this table; but these contents are filled when the process goes to that phase.

Category	Security requirement			
Number				
Name				
Source	Module	Threat category	Threat name, priority	Abuse case number
High level security requirements				
Derived functional security requirements				

Figure 12: SHS security requirements table

4.3.3 Deriving functional security requirements from high level security requirements

This phase aims at deriving the functional security requirements from the high level security requirements. Ideally, the implementations of those functional requirements in the system would fulfill the high level security requirements. Three steps are needed to derive the functional security requirements and they are depicted in Figure 13.

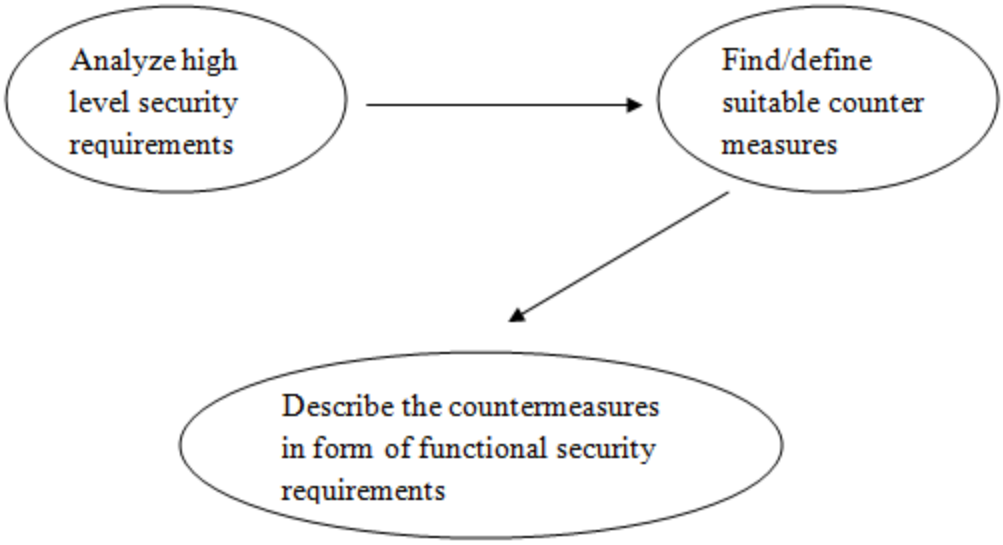

Figure 13: Steps to derive functional security requirements for the SHS under development

4.3.3.1 Analyze high level security requirements

This step is to find some clues for finding/defining suitable countermeasures to fulfill high level security requirements. The abuse cases can be used to assist this step because they describe how the security threats can be carried out.

4.3.3.2 Identify/define suitable countermeasures

In this step proper countermeasures are identified or defined to fulfill high level security requirements.

Many of the security threats imposed on the information systems had been analyzed in great detail and a lot of countermeasures are proposed to deal with them. As a result, for a particular high level security requirement, the task is to find the suitable countermeasures and tailor it for the system under development.

Sometimes, due to some domain/application specific constraints, there are no available countermeasures and as a result, new countermeasures should be defined to resolve the security threats.

4.3.3.3 Describe countermeasures in form of functional requirements

In this step, the identified/defined countermeasures are specified in the form of functional security requirements and they are recorded into the derived functional security requirements section of the security requirements table.

4.4 Summary

This chapter introduces a systematic method to generate security requirements for the SHS. The main outputs of this method are the system security threats profile and a collection of system security requirements specifications.

Chapter 5

CASE STUDY

The approach proposed in chapter 4 is applied to an imaginary SHS. The purpose of this chapter is to give an example on how to use the proposed approach to generate the SHS security requirements specifications.

5.1 The imaginary SHS:

5.1.1 System functionalities and system structure

Many practical SHS are researched for their functionalities, structures and user interfaces when the imaginary SHS is being defined. As a result, the functionalities of the imaginary SHS are common among the SHS seen in the market. The functionalities of the imaginary SHS are lighting control, HVAC and home safety/security. The system structure confirms to one described in the general abstract SHS system model (Figure 4).

Each of the functionalities is implemented by an individual module and these modules communicate with each other through the wireless communication links.

There is a specially designed SHS controlling software runs on the mobile tablet in this imaginary SHS. A user can use that software to control and communicate with the SHS remotely. User commands are sent to the host controller and then the host controller passes these commands to related modules. System status can be obtained and displayed on the mobile tablet upon user request or in other user definable manner. This mobile tablet corresponds to the remote node as discussed in chapter 4.

It is possible for this SHS to be controlled by other kind of remote node but for simplicity it is assumed the mobile tablet is the only remote node in this SHS.

5.2 The applying of the proposed solution

The stakeholders' security needs for the imaginary SHS can be expressed as simple as a single sentence: "The newly developed SHS should be secure"; and this is the start point for the security requirements development process.

The purpose of this case study is for illustration; as a result only the security requirements for the host controller will be generated by using the proposed method.

5.2.1 Creating the SHS security threats profile

5.2.1.1 Abstract system model for the imaginary SHS

The abstract system model of the imaginary SHS which is shown in figure 14 is an instance of the general abstract SHS system model that is discussed in chapter 4. This is because its structure confirms to the structure of it.

.

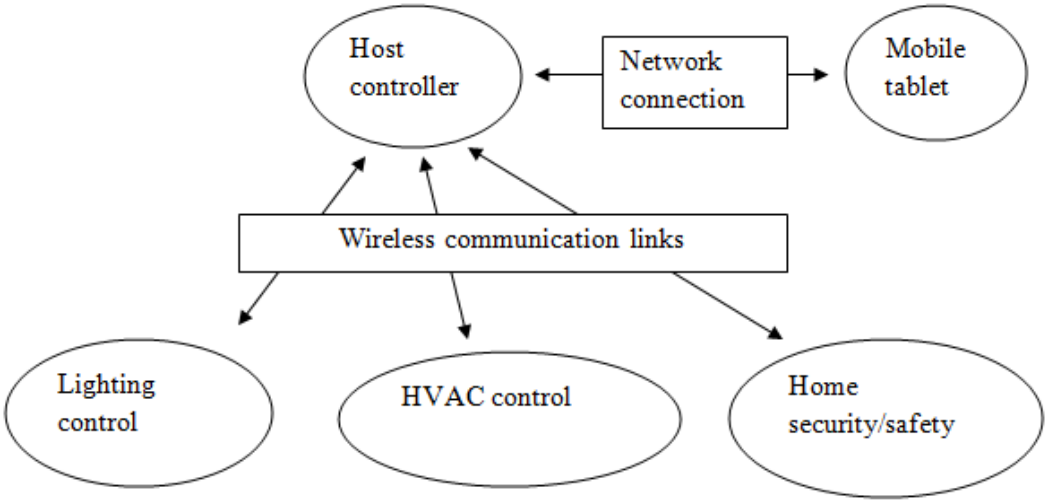

Figure 14: Abstract system model for the imaginary SHS

5.2.1.2 Identify security assets and security threats

The first step in this process is to build the nodes information table. The building of it is straight forward because the information needed can be obtained by reading other system design/development documents. In this case study, only the information about the host controller is collected because this case study is intended to generate the security requirements for host controller only. The collected information is shown in Figure 15.

Node	Mobile tablet	Host controller	Lighting control	HVAC control	Home security/safety
User interface		A touch panel for local control; a network port for receiving remote commands and reporting system status.			
Functionalities		Receiving remote commands to perform actions; collecting sub nodes status and reporting them to mobile tablet; control sub nodes by the commands entered by local users through touch panel			
Communication links		Communicate with mobile tablet via network connection; communicate with sub nodes via internal wireless communication link			
Data on node		User credential; system configuration data; system status logging data			
Other					

Figure 15: Nodes information table for the imaginary SHS

It is assumed, in this case study, a security threats check list is not available when the SHS is developed and the system developers are required to create the check list.

This check list is created by using the conception and template discussed in section 3. The created security threats check list is assumed to be complete for the illustration purpose and the checklist is shown in Figure 16.

Threats category	Threat name	Description
Network	Eavesdropping	Try to get critical data transferred on network by eavesdropping without being noticed by the legal system user.
	Spoofing	Try to access the system by duplicating the credential of a legal user.
Internal communication link	Eavesdropping	Try to get critical data transferred on internal communication links by eavesdropping without being noticed by the legal system user.
	Spoofing	Try to access a node by duplicating the credential of a legal user.
Node	Unauthorized data access	Try to read, delete and modify critical data stored in a node.
	Application code attack	Try to make the node totally unworkable by erasing the application code in it.

Figure 16: The security threats check list for the imaginary SHS

The next step is to identify security assets and security threats they are facing after the nodes information table and the security threats check list are ready.

The host controller is evaluated against each of the security threats listed in the check list. The node information collected for the host controller is used during this evaluation process.

After the evaluation, the host controller is found to be facing the security threats of network eavesdropping and spoofing, internal communication link

eavesdropping and spoofing and unauthorized data access. These threats are marked out in the security assets and threats table (Figure 17).

Threat Category	Threat name	…	Host controller	Priority	…
Network	Eavesdropping		Y	1.5	
	Spoofing		Y	1.8	
Internal communication link	Eavesdropping		Y	1.5	
	Spoofing		Y	1.8	
Node	Unauthorized data access		Y	2.67	
	Application code attack				

Figure 17: The security assets and threats table of the imaginary SHS

5.2.1.3 Defining abuse case:

For each of the security threats the host controller faces an abuse case is defined. One of the purposes of this step is to get a better understanding on how a particular security threat can be carried out; another purpose is to generate the possibility and risk level information which is used to calculate the priority of a particular security threat. The possibility and risk level information are obtained by following the steps defined in chapter 4.

These abuse cases are shown from Figure 18 to Figure 22.

Abuse case number	1			
Abuse case name	Eavesdropping the data transferred between host controller and mobile tablet over the network.			
Source	**Node**	**Threat category**		**Threat name**
	Host controller	Network		Eavesdropping
Purpose	Try to get critical information, like user name/password, contained in the network data packages by eavesdropping so that this activity is not notified by the system user			
Steps				
1	The attacker sets up software and special devices to gather the data packages transferred between mobile tablet and host controller over the network.			
2	The attacker gathers enough network data packages and tries to interpret the content in them.			
3	The content in the network data package is interpreted by the attacker. He/she knows where the user name/password portion is and where data portion is in the network data package.			
Consequence	The attacker can build a valid but illegal network data package to sent to the host controller			
Possibility criteria	Skill Level	Time Cost	Money Cost	Average
Score	2	2	2	2
Possibility	0.5			
Risk level criteria	Damage potential	Reproducibility	Number of user affects	Average
Score	3	3	3	3
Risk level	3			

Figure 18: The abuse use for networking eavesdropping security threat on the host controller

Abuse case number	2			
Abuse case name	Spoof the host controller by sending valid but illegal network data package to it over the network			
Source	**Node**	**Threat category**	**Threat name**	
	Host controller	Network	Spoofing	
Purpose	Try to make the host controller to perform certain action			
Steps				
1	The attacker constructs a valid but illegal network data package by using user name/password of a legal system user and other information obtained by using the eavesdropping attack.			
2	The attacker sends the network data package to the host controller over the network			
3	The host controller evaluates the network data package it received and finds it is a valid data package.			
4	The host performs the action according to the command contained in the data package.			
Consequence	The host may perform an action based on the command given by an attack. For example the action is turning off the security/safety module in the SHS			
Possibility criteria	Skill Level	Time Cost	Money Cost	Average
Score	2	1	2	1.67
Possibility	0.6			
Risk level criteria	Damage potential	Reproducibility	Number of user affects	Average
Score	3	3	3	3
Risk level	3			

Figure 19: The abuse case for network spoofing security threat on the host controller

Abuse case number	3			
Abuse case name	Eavesdropping the data transferred between host controller and nodes over the internal communication link			
Source	**Node**	**Threat category**	**Threat name**	
	Host controller	Internal communication link	Eavesdropping	
Purpose	Try to get critical information, like user name/password, contained in the data packages transferred on the internal communication link by eavesdropping so that this activity is not notified by the system user			
Steps				
1	The attacker sets up the device and software to monitoring wireless communication around the SHS system.			
2	The attacker gathers enough wireless communication data package and tries to interpret them.			
3	The content in the wireless communication data package is interpreted by the attacker. He/she knows the meaning of data contained in the wireless communication data package.			
Consequence	The attacker can construct valid but illegal wireless communication data packages to be sent to the host controller in the SHS.			
Possibility criteria	Skill Level	Time Cost	Money Cost	Average
Score	2	2	2	2
Possibility	0.5			
Risk level criteria	Damage potential	Reproducibility	Number of user affects	Average
Score	3	3	3	3
Risk level	3			

Figure 19: The abuse case for the internal communication eavesdropping security threat on the host controller

Abuse case number	4			
Abuse case name	Spoofing			
Source	**Node**	**Threat category**		**Threat name**
	Host controller	Internal communication link		Spoofing
Purpose	Spoof the host controller by sending valid but illegal network data package to it over network			
Steps				
1	The attacker constructs a valid but illegal wireless communication data package by using their knowledge about the wireless communication data package.			
2	The attacker sends the wireless communication data package to the host controller over the wireless communication links.			
3	The host controller evaluates the wireless communication data package it received and finds it is a valid data package.			
4	The host believes the information contained in the wireless communication data package.			
Consequence	The attacker can spoof the host controller about status of a sub node in the SHS. For example the attacker can construct wireless communication data packages saying that the HVAC system is turned on while actually it is turned off.			
Possibility criteria	Skill Level	Time Cost	Money Cost	Average
Score	2	1	2	1.67
Possibility	0.6			
Risk level criteria	Damage potential	Reproducibility	Number of user affects	Average
Score	3	3	3	3
Risk level	3			

Figure 20: The abuse case for internal communication link spoofing security threat on the host controller.

Abuse case number	5			
Abuse case name	Modify critical system data			
Source	**Node**	**Threat category**	**Threat name**	
	Host controller	Node	Unauthorized data access	
Purpose	Try to read, delete and modify critical data stored in the host controller.			
Steps				
1	The attacker knows how to log into the system as a guest user.			
2	The attacker logs into the system using the user name/ password of a guest user.			
3	The attacker goes to system configuration section.			
4	The attacker reads and modifies system configuration data.			
Consequence	The attacker can read and modifier the critical system data at his/her own will			
Possibility criteria	Skill Level	Time Cost	Money Cost	Average
Score	1	1	1	1
Possibility	1			
Risk level criteria	Damage potential	Reproducibility	Number of user affects	Average
Score	2	3	3	2.67
Risk level	2.67			

Figure 21: The abuse case for the node unauthorized data access security threat on the host controller

5.2.1.4 Prioritization

Prioritization is a straight forward process. The priority is obtained by using the algorithm defined in chapter 4 and the calculated priority is recorded into the security assets and threats table for the imaginary SHS.

5.2.2 Defining high level security requirements.

The process for defining the high level security requirements can be carried out under the assistance of all the information obtained in the previous steps.

The security threats with higher priority value are considered first and for each of them a high level security requirement is defined.

For the security threats with very low priority value, the system developers may chose to accept such threats to happen due to cost, time - to- market constraints.

The high level security requirements for the host controller in the imaginary SHS are recorded in the system security requirements table. Figure 23 – figure 27 shows the defined high level security requirements.

Category	Security requirement			
Number	1			
Name	Security requirement for preventing the host controller from unauthorized data access security threat			
Source	**Module**	**Threat category**	**Threat name, priority**	**Abuse case number**
	Host controller	Node	Unauthorized data access,2.67	5
High level security requirements	After a user logs into the host controller, the user can only access the data, stored in the host controller, that is allowed to be accessed by the user.			
Derived functional security requirements	1. When a user tries to access a data stored in the host controller, after he/she logged into the host controller, the user's privilege should be checked. 2. If the user has the right privilege then he/she is allowed to access the data.			

Figure 23: Security requirement for preventing the host controller from unauthorized data access security threat

Category	Security requirement			
Number	2			
Name	Security requirement for preventing the host controller from network spoofing security threat			
Source	**Module**	**Threat category**	**Threat name, priority**	**Abuse case number**
	Host controller	Network	Spoofing, 1.8	2
High level security requirements	The host controller should be protected from being spoofed by network data packages.			
Derived functional security requirements	1. After the host controller received the network data package, if it is encrypted, the host controller should turn it into plan data. 2. The host controller should authenticate the identity information contained in the plan data before performing other actions.			

Figure 24: Security requirement for preventing the host controller from network spoofing security threat

Category	Security requirement			
Number	3			
Name	Security requirement for preventing the host controller from internal communication link spoofing security threat			
Source	**Module**	**Threat category**	**Threat name, priority**	**Abuse case number**
	Host controller	Internal communication link	Spoofing, 1.8	4
High level security requirements	The host controller should be protected from being spoofed by internal wireless communication data packages.			
Derived functional security requirements	1. After the host controller received the internal wireless communication data package, if it is encrypted, the host controller should turn it into plan data. 2. The host controller should authenticate the identity information contained in the plan data before performing other actions.			

Figure 22: Security requirement for preventing the host controller from internal communication spoofing security threat

Category	Security requirement			
Number	4			
Name	Security requirement for preventing the host controller from network eavesdropping security threat			
Source	**Module**	**Threat category**	**Threat name, priority**	**Abuse case number**
	Host controller	Network	Eavesdropping, 1.5	1
High level security requirements	The data transferred between the host controller and the mobile tablet over the network should be protected from eavesdropping.			
Derived functional security requirements	1. The data in the network data package should be encrypted before the mobile tablets sending them out to the host controller over the network. 2. The data in network data package should be encrypted before the host controller sending it out to mobile tablet over the network.			

Figure 236: Security requirement for preventing the host controller from network eavesdropping security threat.

Category	Security requirement			
Number	5			
Name	Security requirement for preventing the host controller from internal communication link eavesdropping security threat			
Source	**Module**	**Threat category**	**Threat name, priority**	**Abuse case number**
	Host controller	internal communication link	Eavesdropping, 1.5	3
High level security requirements	The data transferred between the host controller and other system sub modules over the internal communication link should be protected from eavesdropping.			
Derived functional security requirements	1. The data in the wireless communication data package should be encrypted before the system sub nodes sending it out to the host controller over the internal wireless communication network. 2. The data in the wireless communication data package should be encrypted before the host controller sending it out to a system sub node over the internal wireless communication network.			

Figure 247: Security requirement for preventing the host controller from internal communication link eavesdropping security threat

5.2.3 Derive functional security requirements from high level security requirements

After defining the high level security requirements the corresponding functional security requirements can be derived from them. For example, for the high level security requirement to prevent host controller from unauthorized data access security threat, a privilege based method can be used to fulfill this requirement. As a result, the derived functional requirement could be:

When a user tries to access a data stored in the host controller, after he/she logged into the host controller, the user's privilege should be checked.

If the user has the right privilege then he/she is allowed to access the data.

These two functional security requirements are recorded into the corresponding system security requirements table.

The same process is applied to other high level security requirements. By the end of the process, the security requirements specification for the host controller is generated (Figure 23- Figure 27).

When such a process is applied to all other nodes in the SHS, a complete SHS security requirements specification can be generated.

Chapter 6

CONCLUSION AND FUTURE WORKS

6.1 Conclusion

By noting the importance of the security features of the SHS, the lacking of a systematic method to understand and solve security problems the SHS facing from the requirements engineering level and the problems in security requirements specification, a systematic approach to generate to security requirements for a SHS under development is proposed in this paper.

Briefly, in this approach, a SHS security threats profile is build first and then the high level security requirements and functional security requirements are defined and derived.

The building of the SHS security threats profile is the core part of this solution. It gives the system developers a SHS security threats check list; a clear understanding on what security threats a SHS under development might face; the steps needed to carry out those security threats; their possibilities to happen; their risk level and their priorities. The SHS security threats profile is the basis for defining and deriving security requirements.

This solution can enhance the consistency in the selection of security requirements because the building of the security threats profile forces the system developers to identify the security threats a SHS might face as completed as possible and as a result there is less chance to leave out security requirements on some security relevant areas.

It can also enhance the consistency in the level of specification details because it requires the system developers to define high level security requirements first and then to derive functional security requirements.

6.2 Future works

Although the SHS security requirements specifications can be generated by using the proposed approach, there are still a lot of works could be done in future to complete the SHS security requirements engineering process.

A security threats check list that is specific to the SHS can be build in future. This check list can be used directly or used as a reference by the system developers when they are dealing with the security problems in the SHS.

The functional security requirements generated by this approach may contain ambiguities and the formal specification is an effective way to resolve them. As a result, another future work is to formalize these security requirements by using a kind of formal language.

The specified security requirements should be verified and validated to ensure they are actually fulfilling the stakeholders' expectations. As a result, establishing a security requirements verification/validation method which is specific to the SHS is also a direction of the future work.

REFERENCES

Ambrosio Toval, J. N. (n.d.). Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach.

Charles B. Haley, R. L. (2008, January/Febuary). Security Requirements Engineering: A Framework fo rRepresentation and Analysis. *IEEE TRANSACTIONSON SOFTWARE ENGINEERING, VOL.34, NO.1*, p. 133.

Dale R. Thompson, N. C. (n.d.). RFID SECURITY THREAT MODEL.

EbenezerA.Oladimeji, S. a. (n.d.). SECURITY THREAT MODELING AND ANALYSIS: A GOAL ORIENTED APPROACH.

Elizabeth Hull, K. J. (2011). *Requirements Engineering, 3rd edition.* Springer.

Eržen, R. (2012). Review of main security threats in Smart Home networks.

Georgios Mantas, D. L. (2011). Security in Smart Home Environment.

Goodwin, S. (2010). *Smart Home Automation with Linux, 1st edition.* Apress.

Guttorm Sindre, A. L. (2004). Eliciting security requirements with misuse cases. *Requirements Eng (2005).*

Home automation. (n.d.). Retrieved from Wikipedia: http://en.wikipedia.org/wiki/Home_automation

John McDermott, C. F. (n.d.). Using Abuse Case Models for Security Requirements Analysis.

John Wilander and Jens Gustavsson. (2005). Security Requirements -A Field Study of Current Practice. *Dept.of Computer and Information Science, Link¨opings universite*.

Microsoft, J.D. Meier, Alex Mackman. (n.d.). *Threat modeling.* Retrieved from http://msdn.microsoft.com: http://msdn.microsoft.com/en-us/library/ff648644.aspx

Mohd Ariff Razaly, M. S.-R. (2012). A Review of Security System for Smart Home Applications. *Journal of Computer Science 8 (7, ,* pp. 1165-1170.

Paul Kocher, R. L. (2004). Security as a New Dimension in Embedded System Design. *DAC 2004, June7–11*.

Philip Koopman, Carnegie Mellon University. (2004, 7). Embedded System Security.

Rosslin John Robles, T.-h. K. (2010, February). A Review on Security in Smart Home Development. *International Journal of Advanced Science and Technology Vol. 15*.

Rushby, J. (2001, March). Security Requirements Specifcations: How and What?

Sommerville, I. (2011). , *Software engineering, 9th edition.* Pearson Education.

Suvda Myagmar, A. J. (n.d.). Threat Modeling as a Basis for Security Requirements.

Tanenbaum, A. S. (2007). *Modern operating system, 3rd edition.* Prentice Hal.

Young, R. R. (2004). *The requirement engineering handbook* . Artech House.