Towards a Structural Secure Design Process

S. Hasan Mirjalili
IC, EPFL
Lausanne, Switzerland
Seyydhasan.mirjalili@epfl.ch

Arjen K. Lenstra
IC, LACAL, EPFL
Lausanne, Switzerland
Arjen.Lenstra@epfl.ch

Abstract — **Computing has become part of everyday life by means of consumer electronics, embedded systems and other computing systems. Security as a new dimension in the design of computing systems has increased the complexity of design process. A structural secure design methodology would facilitate the design process and improve the assurance on security of a system. In this paper a structural design process for considering security from the beginning is proposed.**

Keywords— security; design process; embedded systems; design methodology

I. INTRODUCTION

Many devices produced today require security. The assets on the device must remain confidential, available and protected against manipulation according to defined policies. The success of businesses, the satisfaction and confidence of users depend on the strength of the device security. Making secure a device with various assets, different security expectations and multiple stakeholders is difficult. Many parameters should be considered to design a secure device. For example, consider an embedded system, which stores, processes and transmits data of different stakeholders, especially current devices with many features and naturally various security requirements. Due to the complexity of the system, if there is no systematic analysis and design method, the assets on the system might be weakly secured or not be secured at all. Security breaches are not always by breaking cryptographic algorithms or using technical methods. Ross Anderson in his survey of the failure modes of secure systems titled "why cryptosystems fail" [1] discusses that most frauds were not caused by cryptanalysis or other technical attacks, but by implementation errors and management failures.

Clearly, it is important to secure systematically a computing system against security breaches. It is unwise to start building a house, for example, without first considering a design, which provides a secure and safe house, and the requirements the house should fulfill. This idea applies to securing computing systems too: careful planning ensures the implementation of optimal and proper safeguards and controls in a system. Therefore one needs to utilize a design method that considers security from the beginning and throughout the life cycle of the system.

Secure system design methods have evolved in computer security history from checklists and standards to secure design methodologies. The idea of checklists is to identify possible countermeasures and to pick out the security solutions, which are needed from this list. Security standards such as ISO/IEC 17799 (subsequently renumbered ISO/IEC 27002) [2], and GAISP (Generally Accepted Information Security Principals) [3] followed after checklists. A difference between checklists and standards is that the standards attempt to establish international, authoritative, and generic security criteria. In response to the limitations of security checklists and standards, security methods appeared. In contrast to standards and checklists, these methods begin development by exploring systems security requirements. The methodologies can be classified into five categories: stepwise; object-oriented; viable; information modeling; and responsibility modeling. Some of these methods take influence from previous information systems (IS) and software development methods.

Available methods are mostly either for IS in organizations or software development. However, nowadays there are many computing devices with complicated life cycle and numerous features, for example: smart phones, GPS systems, medical devices, and so on. To design such devices, the methodologies for secure IS in organizations or software design methodologies are not the solution; however, they can be adapted. Life cycle of assets on electronic devices is more complicated than software systems or IS. Also in electronic devices, in some scenarios, there are many entities involved throughout the life cycle of the system and this makes the security of system more challenging than software systems or IS.

The first step in the design of a secure system is to specify the security requirements of the system. Existence of a systematic and structural method for specifying the security requirements of a system is significantly helpful. It helps to have a level of assurance that the essential security requirements were not neglected. If this method is structural and step by step, it also reduces the complexity of design. Security requirement specification is to specify the functional and non-functional requirements and policies that a design should have in order to be considered secure and the assets on the system are protected. A structural design method has many advantages that are emphasized here:

- In most cases, a system is designed then based on the realized system the security engineers play the role of an adversary and try to have a threat model for the system. The output of this threat modeling is a set of security requirements. In these cases they may not have a systematic and holistic method for the specification of the requirements and they are not sure about the completeness of the system functionalities. After releasing the product, they may find that the system needed some additional features and it may be late because there may not be more available resources on the system or may lead to a fundamental change in the design of the system. However, a structural method simplifies the study of the system life cycle in order to list its security requirements, and to manage and prioritize them and have provisions for all the security expectations. For example, a company may design a device, which stores data of the user securely

however the transfer of data from one device to another, in case the user bought a new device, might not have been foreseen. Later the designers would have difficulty in adding this feature to the device because there are limited resources on the device for new feature while this could be predicted before releasing the device. If there was a systematic method and they had a holistic view of the system from the beginning throughout the entire life cycle, then the designers could manage the resources and prioritize the requirements.

- Considering security sooner in design process means less unnecessary costs for the stakeholders. There is a study [4] that shows the sooner vulnerabilities are detected the less expensive is fixing them.
- A structural requirement specification and design process simplifies the process and they need not to have extensive experience in security engineering. A designer with basic knowledge of security can follow the steps and reach to a set of security requirements for his design.
- A well developed design method can be automated and help the designer to computerize the task of requirement specification and other stages of the design.
- A design method can also be used for analysis and evaluation of an off the shelf product.
- A security requirement method can act as a modeling and common language between designers and implementers, also as a reference for new designs or modifications.
- It is also useful as a model for study of future changes and their impacts on the entire system.

Considering all the advantages mentioned above as motivation for our work, in this paper a structural security design method is proposed.

In Section II a review of related work is presented. Most of the design methods are related to IS and some works are proposing specific secure design methods for specific security problems. In Section III, the structural security design methodology that is main contribution of this paper is described in detail. The last section is the conclusion of the paper where the findings of paper are summarized.

II. RELATED WORK

Earlier review studies [5][6][7] have classified IS design methods into five paradigms: security-modified IS design methods, responsibility method, business process methods, information modeling methods, and viable and survivable system methods.

1. Security-modified IS design paradigm

The security-modified IS design paradigm are secure aware version of IS or software development methods. This paradigm includes methods such as the logical control method [8], the spiral model for secure IS design [9], virtual methodology [10] [11], the soft method for the planning of secure IS [12], meta-notation [13], secure IS planning methodology [14] and UMLsec [15], security design patterns [16] and integrated security and systems engineering approach [17][18].

The *logical control method* adds the required security controls into processes. Threat classes and corresponding controls are listed in a data dictionary.

The *spiral model* for secure IS design integrates security concerns into Boehm's spiral model [19].

Virtual methodology is modified version of soft systems methodology [20], emphasizing organizational, cultural and human factors in the development and design of IS design.

The *soft method for planning of secure IS* is also based on the soft system methodology. This method encourages the role of the user contribution in the design and development of secure IS to exploit the user's knowledge of the organization activities and to increase user's awareness and commitment to the designed system.

Meta-notation like the logical control method and the spiral model adapts IS and software development methods. In this method six security elements are proposed to be integrated into IS and software development methods to make those methods secure aware.

Secure IS planning methodology is composed of three components: security risk planning model, awareness program, and countermeasure matrix. This method proposes five stages for secure IS planning: recognition of security problems; risk analysis; generation of alternatives; decision making; implementation. The awareness program and countermeasure matrix are integrated into these stages.

UMLsec is security extension of UML by proposing new security elements like {tags} and <<stereotypes>> to UML. It allows expressing security requirements in a system specification.

Security design patterns address the security in systems design using best practice solutions to show how to integrate security in the engineering process. This methodology offers a pattern catalog, which enables designers to pick security patterns, which meet their particular requirements.

The *integrated security and systems engineering approach* extends Tropos method with security features. Tropos is a software development methodology, where concepts of the agent paradigm are used along the entire software development process [21].

2. Responsibility modeling paradigm

Secure IS design methods in the responsibility modeling paradigm assume security requirement of a system can be extracted by investigating the job responsibilities in organizations. Examples of these methods are abuse-case-based [22], eliciting security requirements with misuse cases [23] the task-based analysis method for a better authorization model [24]. According to [25] security requirements, i.e., need-to-do and need-to-know, can be drawn from the job responsibilities.

3. Business process paradigm

The business process paradigm consists of methods such as providing security semantics in workflow management [26], the fair and secure electronic markets model and infrastructure [27]. These methods aim to build a modeling notation for describing security constraints in business process models.

4. Information modeling paradigm

The information modeling paradigm includes methods attempt to present security notations particularly for developing secure databases. Security constraints during multilevel secure database design method [28], object-oriented modeling of security semantics [29], modeling data secrecy and integrity with data flow diagrams and entity-relationship [30] and OLAP (On-Line Analytical Processing) Security Design [31].

5. Viable and survivable IS paradigm

The viable and survivable IS paradigm includes methods that resist attacks and are survivable (methods to build systems to cope with present and future attacks). [32] is a viable system model and [33] a survivable system design method.

III. STRUCTURAL SECURITY DESIGN METHODOLOGY

After mentioning the motivation behind a structural security design method and listing other design methodology related to this paper, in this section the proposed method is explained. The foundation of the method is based on this idea that for securing a system, the principal is protection of assets in a system based on defined policies. Therefore, in securing a system 3 phases as it is shown in Figure 1 are followed:

A. Identifying assets
B. Defining policies for securing each asset
C. Enforcing the policies on the assets

To make an entire system secure, the set of assets on the system should be protected. For each asset, these 3 phases should be followed. Each phase consists of one or several steps. The outline of the Structural Security Design Process is as follows:

A. Identifying assets
- Step 1. Identification of resources
- Step 2. Security properties specification
- Step 3. The entities involved in the system lifecycle
- Step 4. The life cycle of the system

B. Defining policies for securing each asset
- Step 5. Defining policies

C. Enforcing the policies on the assets
- Step 6. Security Strategy
- Step 7. The acquired technology
- Step 8. Risk Assessment

Each step is explained in detail in the next subsections. Note that when we refer to "designer", it may imply that a single person is responsible for the design process, but in most cases, a cross section of the entire corporation (technical and business people) forms the team of design. Determining efficient composition of the design team and responsibility of each member of the team is important but it is out of the scope of this paper.

A. Identifying assets

Before starting identification of assets in a system, the system must have been defined. Features, data, processes, and functionalities should have been specified without security considerations. Out of this information from the system, a designer(s) can identify the assets on the system. In fact, s/he knows what the system is and wants to make it secure. A designer just follows the steps to make the specified system secure.

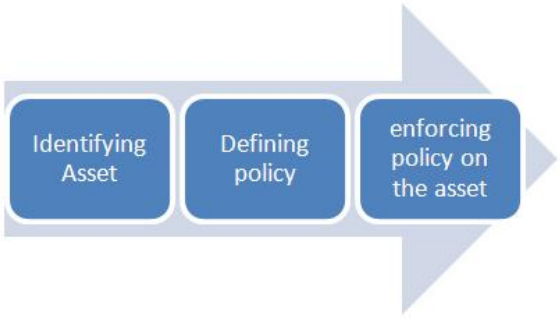

Figure 1-The process of securing an asset in a system

In a system, an asset is a resource, which needs security properties and consequently protection. Among the resources, some need security and some do not need so. Those which need security are identified as assets. These resources are generally valuable and sensitive. Security properties means the assets need one or multiple of the security properties. These properties can be, for instance, CIA triad (Confidentiality, Integrity, and Availability) or Parkerian Hexad (will be explained in Step 2), or other properties such as: untraceability, anonymity, uncloneability, etc.

An asset has 4 aspects. First an asset is a resource in a system. Second the resource has to require security property to be considered asset. A resource without security requirement is not considered an asset. Third aspect is that there should be some stakeholders and adversary for that asset, otherwise if a resource is not valuable and/or accessible to anybody but his owner, then it is not considered an asset, which needs protection. Forth aspect is the life-cycle of the asset. As it is shown in Figure 2, to identify assets in a system and to define their security policy, four aspects of the asset (resources, security properties, life cycle of the resources, and entities involved throughout the life cycle of the resources) should be specified. As it is shown in Figure 3, the designer can start from listing resources and continue to determining security property of the resources, studying the life cycle of the resource and then detecting the entities involved in the life cycle of the resource. Of course finishing one step does not mean that the step is complete and should not be revised that step again. For example, after listing all resources and during listing entities, one may notice resource(s), which have been neglected in the previous step. In these circumstances, the list of resources is updated and the next steps are reviewed. These circular steps are taken and retaken until the designer believes that steps are complete and nothing has been neglected.

Step 1. Identification of resources

All resources are identified, no matter if they need to be protected or not. The resources, which do not require security considerations are not the main concern of a secure design, but they should be identified and documented. There are two types of resources in a system: Data and process. Data refers to a

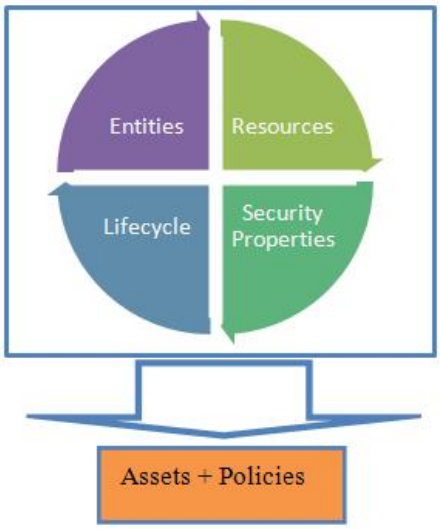

Figure 2- The process of identifying assets in a system

collection of numbers, characters, images or other outputs from system to convert physical quantities into symbols. Such data is typically further processed by a human or input into a computer, stored and processed there, or transmitted (output) to another human or computer (possibly through a data connection). Pictures taken and stored on a device, contact list on a mobile phone are example of data. A process is an instance of a computer program that its instructions are being sequentially executed by a computer system to fulfill functionality or service. For instance, calculator program on a mobile phone is a process.

Step 2. Security properties specification

After listing the resources on the system, for each resource, we should specify the security properties the resource requires. In the literature various set of security properties have been proposed. The most accepted and generic is CIA triad: Confidentiality, Integrity and Availability [34].

Another approach with additional security properties is Parkerian hexad [35]; a set of six information security properties proposed by Donn B. Parker. The Parkerian hexad adds three additional properties to the three classic security properties of the CIA triad: *Possession or Control*, *Authenticity*, *and Utility*.

Possession or Control: Suppose a thief were to steal a sealed envelope containing a bank debit card and (foolishly) its personal identification number (PIN code). Even if the thief did not open that envelope, the victim of the theft would legitimately be concerned that s/he could do so at any time without the control of the owner. That situation illustrates a loss of control or possession of information but does not involve the breach of confidentiality. *Authenticity* refers to the veracity of the claim of origin or authorship of the information. *Utility* means usefulness. For example, suppose someone encrypted data on disk to prevent unauthorized access or undetected modifications – and then lost the decryption key: that would be a breach of utility. The data would be confidential, controlled, integral, authentic, and available, but they just wouldn't be useful in that form.

The designer selects the set of security properties corresponding to the system, the environment where the system is used and other conditions like legal issues or standards that enforce a security property on an asset. The outputs of this step are the set of assets associated with the security considerations determined for each asset.

Step 3. The entities

Anyone who involves in the life cycle of the system and has possibility to access asset(s) (protected or unprotected) has to be identified. In some cases different entities have their assets on the same system with different security needs and our design should satisfy the interest of all entities having asset(s) on the system. Another way of specifying entities is the identification of entities who are involved in the processing of resources. We identify the entities based on the action they do on the resources during the life cycle of the system. For example, we identify the entities that do one of actions such as create, edit, copy, transmit, disable, enable, and so on.

Step 4. The life cycle of the system

The stages of a system life cycle changes when the status of assets or the entities are changed. The security requirement of an asset may change during its life cycle. A designer should have prediction and vision about different stages of system life cycle to prevent the possible vulnerabilities in different situations. General life cycle of a system is typically Birth, Life, and Death. However each phase of life cycle is divided to shorter stages. For example, Birth is composed of requirement specification, design, implementation, testing, and so on. Although many systems have similar life cycle but every system has its own specific life cycle. It is the task of designer to study the life cycle of the concerned system.

B. Defining policies

Corresponding to the assets identified in the previous steps and their associated security properties, the policies are defined in this step.

Step 5. Defining policies

At this step we have a model of the system. We have a clear understanding of assets, their required security properties, the life cycle of the assets, and involved entities or stakeholders. Based on this model we define the security policies, which should be applied to the system. Security policy is defined for each asset separately, although we may group some assets and apply same policy on them. The security policy addresses what (asset) should be or should not be accessed by whom (entities) and when (life cycle).

C. Enforcing the policies on the assets

In the first five steps, the assets are identified and policies to be enforced are defined. In fact we have accomplished two first phases illustrated in Figure 1; identifying assets and defining policies. Then corresponding to this system model, we decide on the security requirements and protection mechanisms.

Step 6. Security Strategy

We can have different strategies to tackle the security problem. Each strategy offers different level of security and certainly different amount of efforts and investment (temporal and monetary). In the literature different classifications are proposed. In [36] a four level security strategy has been proposed. The proposed strategy is *Prevention*, *Tolerance*, *Removal* and *Forecasting*. Donn Parker proposes 6 levels, which are *Avoidance*, *Deterrence*, *Prevention*, *Detection*, *Recovery* and *Correction* [37].

Prevention is the first and the best strategy to create a secure system. It means preventing the occurrence or introduction of vulnerabilities. Mostly this is done by solutions and techniques during the development of the system. Improvement of design and development methods can result in good strategies for preventing security vulnerabilities. Prevention is the strategy of recognizing the threats and vulnerability of the system and implementing the corresponding solution to prevent an attack to occur. For example encryption is a prevention strategy.

Tolerance means providing service in spite of some vulnerabilities or faults in the system. Different techniques can be applied towards tolerance such as *Recovery* and *Self-adaptive (self-healing)* techniques. Depending on the attack, different strategies can be followed. Attacks on confidentiality may not be recoverable although they may be prevented in future designs. For attacks on data integrity, we can for example replace the data from the backup. The usual recovery mechanism for attacks on availability is redundancy. For integrity and availability, techniques such as Roll-back can be applied.

Vulnerability removal or simply *removal* is performed both during the development phase and during the operational life of a system. Vulnerability removal during the development phase of a system life-cycle consists of three steps: verification, diagnosis, correction. Verification is the process of checking whether the system adheres to given properties. If it does not, the other two steps follow: diagnosing the vulnerabilities that prevented the verification conditions from being fulfilled, and then performing the necessary corrections. After correction, the verification process should be repeated in order to check that vulnerability removal had no undesired consequences.

Vulnerability forecasting is conducted by performing an evaluation of the system behavior with respect to attack occurrence. Evaluation has two aspects: qualitative and quantitative. Qualitative evaluation aims to identify, classify, rank attacks, or the event combinations that would lead to system attack. Quantitative or probabilistic evaluation aims to evaluate likelihood that a fault will exist or measuring the difficulty of an attack.

As mentioned before, another example of security strategy proposes 6 levels: *Avoidance*, *Deterrence*, *Prevention*, *Detection*, *Recovery* and *Correction* [37].

Avoidance is in fact risk avoidance, consists of analyzing the system for the features and functions that have inherently too high risk, e.g., sharing resources of system with other

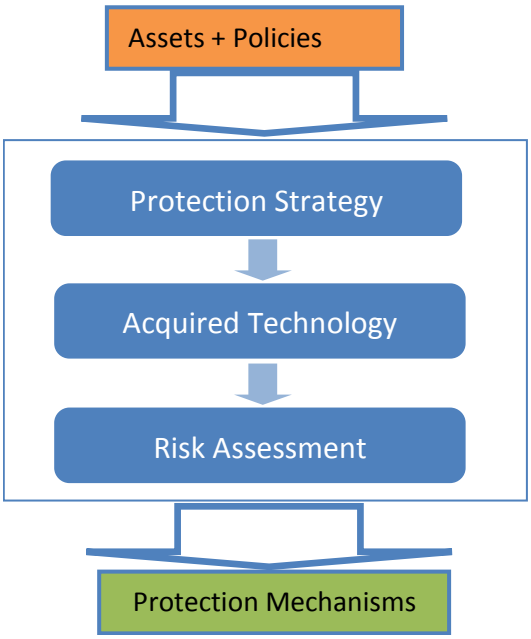

Figure 3- The process of asset protection mechanism based on defined policies

systems or exposing data or processes or providing insecure interface to other systems or users. The need for security and the cost of securing the system may cause these features or functions of the system removed to avoid many risks. Thus, first step is to avoid a feature of system if it causes a high risk for system and it is expensive to make it secure.

Deterrence means to discourage adversary from attack by fear or consideration of dangerous, difficult, or unpleasant consequences. The fear, for example, can be the fear of unanticipated detection. Thus, possible sources of fear are good strategies for deterrence. A simple alarm or a notice on the device can be a deterrence technique.

Prevention is most valued and traditional security safeguard. Computer security comprises mainly preventive measures. Prevention can be absolute, relative or partial. Absolute prevention means that any adversary would fail in any attempt to accomplish his goals. Absolute prevention is only a theoretical concept. A more practical prevention is to force potential adversary to seek other means to accomplish his goals. This is called relative prevention. In partial prevention, the action of adversary is delayed toward his goal.

Detection will make prevention measures more effective. Detection alone is usually insufficient. It is the process of determining that an unauthorized act is impeding, is in progress, or has occurred recently to prevent or to limit losses. Detection also provides a means of recording evidence to determine the cause and source of a detected or potential loss. Password system is a prevention technique for access control and recording failed access attempts is a detection strategy.

If deterrence, prevention and detection are not practical or are not totally effective in dealing with security vulnerability, then *recovery* and *correction* are essential. Recovery can be from a hardware failure, software bugs, operator error or data errors. Check-point and rollback and other methods are in this

category. Correction should be done immediately after recovery.

In summary, if a designer can avoid implementing a feature in the system, he must do it in the first place. Then deterrence discourages attacker to take an action, but if he decided to attack and acted toward violating security of the system, prevention prevents him from achieving his goals. If attacker was successful and prevention techniques were not effective enough, detection can be a method for detecting the cause, preventing future attacks or limiting the losses. Recovery is a solution to return the system to working state and correction is removing vulnerabilities.

Step 7. The acquired technology

We have the choice of various technologies for enforcing policies and security strategies. For example, if we want to secure the confidentiality of an asset, we can use symmetric or asymmetric encryption, or for secure storage we have different technologies with different level of security and of course different cost of investment. In this step the design team investigates the different available technologies corresponding to the system model, selected strategy, and defined policies.

Step 8. Risk Assessment

At this step we have a set of technologies and limited budget and resources to implement the system. This is the step to decide, which functions could be implemented. Cost-benefit analysis is an important component in any design method. It is also known as risk analysis or risk assessment. If the cost of protecting an asset is more than value of the asset or the cost of loss of asset, it is not reasonable to invest on the security of asset. Security in any system should be corresponding to its risks. However, the process to determine, which security controls are appropriate and cost effective is quite often a complex and sometimes a subjective matter. One of the prime functions of security risk analysis is to put this process onto a more objective basis.

The approaches to risk analysis essentially break down into two types: quantitative and qualitative.

A quantitative method of risk assessment proposed by Robert Courtney [38]. It is based on estimates of the expected frequency and amount of loss from each particular realized threat. In Courtney's formula, the expected frequency of threats occurring per year P is calculated (in most cases approximately) by the formula

$$P = 10^{(p-4)}$$

where p is assigned one of the values in Table 1

p=0	If particularly never
p=1	If once in 1000 years
p=2	If once in 100 years
p=3	If once in ten years (taken to be 1000 days)
p=4	If once in a year
p=5	If once a month (10 times a year)
p=6	If twice a week (100 times a year)
p=7	If three times a day (1000 times a year)

Table 1 - Frequency of threats

For example, If frequency of threat is once in thousand years P = 0.001, once in a year P=1, once a month P=10. This numerical method of calculating frequency of threats with the explicit calculations of the dollar loss per event identified as V, yields E, the expected loss per year based on frequency and size of the loss, is given by

$$E = P \times V$$

In many cases it is difficult to determine meaningful probabilities of losses or the amount loss in absolute terms such as dollars in risk assessment formulas. In some cases there is enough recorded experience of incidents (for example, credit card fraud). In most cases however, data are not available and guessing does not produce enough consistent results to induce confidence. In some cases, losses are indirect, such as reputational losses, possibly the most important category, as it affects the future of the asset owner. Therefore quantitative method may provide useful information, but the resulting numbers may be deceiving because of the high degree of guesswork in producing the data. It also entails so much work; hence it should be used for only the most sensitive assets that involve a high potential loss. A detailed quantitative risk assessment can be costly and take a considerable period of time, but some factors like requirements imposed by law, regulation or pressures from external auditors, or pressures from competitors justify the time and investment.

For more practical purposes, it may be more practical to simply rank the seriousness of threat and sensitivity of assets by means of qualitative grades or reasoned risk assessment. For example, in graded risk assessment, vulnerabilities are ranked in the following order:

Highest frequency and highest loss per incident
Moderate frequency and highest loss per incident
Highest frequency and moderate loss per incident
Lowest frequency and highest loss per incident
Lowest frequency and lowest loss per incident

This ranking may need minor adjustments. For example, threats of very high frequency and moderate loss might be higher than threats of moderate frequency and highest loss.

In cases that precise risk assessment is costly and time consuming for the product, the designer can follow these steps:

- Identify obvious vulnerabilities and known safeguards
- Act on those vulnerabilities no matter what
- Can we evaluate assets? Can we group them together? Can we rank them and evaluate relatively?
- Identify threats, is the list complete? Can be grouped and apply to group of assets? Can be ranked based on frequency of occurrence? Is there level of confidence in the ranking?
- Pair off the group of assets and group of threats to identify vulnerabilities. Identify and evaluate controls protecting assets from threats.

IV. CONCLUSION

In this paper a structural design process methodology, which considers security from the beginning and throughout the system life cycle was proposed. This methodology is easy to

follow for designer(s) who is (are) not necessarily a security engineer. Comparing to other design methodologies, in this method, the life cycle of the system is taken into account. This design methodology is suitable for designing devices with dynamic life cycle and involving many stakeholders having their assets on the device.

REFRENCES

[1] R. Anderson, "Why cryptosystems fail," Proceedings of the 1st ACM conference on Computer and communications security, 1993, pp. 215 - 227.

[2] ISO/IEC 27000 family of standards, available online: http://www.27000.org/, 23/4/2010.

[3] Generally Accepted Information Security Principles (GAISP) Version 3.0, available online: http://all.net/books/standards/GAISP-v30.pdf, 23/4/2010.

[4] S. McConnell, Code Complete, 2d ed. Redmond, WA: Microsoft Press, 2004.

[5] M. Siponen, "Analysis of modern IS security development approaches: towards the next generation of social and adaptable ISS methods," Information and Organization, vol.15, issue 4, 2005, pp. 339–375.

[6] M. Siponen, "Secure-system design methods: evolution and future directions," IT professional, vol. 8, issue 3, 2006, pp. 40–44.

[7] M. Siponen and J. Heikka, "Do secure information system design methods provide adequate modeling support?" Inf. Softw. Technol. 50, 9-10 (Aug. 2008), pp. 1035-1053.

[8] R. Baskerville, "Designing Information Systems Security," John Wiley, Information Systems Series, 1988.

[9] H.A.S. Booysen and J.H.P. Eloff, "A methodology for the development of secure application systems," Proceedings of the 11th IFIP TC11 International Conference on Information Security, 1995.

[10] J. Hitchings, "Achieving an integrated design: the way forward for information security," Proceedings of the IFIP TC11 11th International Conference on Information Security, 1995.

[11] J. Hitchings, "A practical solution to the complex human issues of information security design," Proceedings of the 12th IFIP TC11 International Conference on Information Security, IFIP/SEC'96, 1996.

[12] H.L. James, "Managing information systems security: a soft approach," Proceedings of the Information Systems Conference of New Zealand, IEEE Society Press, 1996.

[13] M. Siponen and R. Baskerville, "A new paradigm for adding security into IS development methods,' J. Eloff, L. Labuschagne, R. von Solms,Dhillon (Eds.), Advances in Information Security Management & Small Systems Security, Kluwer Academic Publishers, MA, pp. 99–111.

[14] D.W. Straub and R.J. Welke, "Coping with systems risk: security planning models for management decision making," MIS Quarterly 22 (4) (1998), pp. 41–64.

[15] J. Jurjens, Secure Systems Development with UML, Springer-Verlag, 2004.

[16] B. Blakley and G. Heath, Members of the open security forum, security design patterns technical guide, The Open Group Security Forum, 2004.

[17] H. Mouratidis, P. Giorgini, G. Manson, and I. Philp, "A natural extension of tropos methodology for modelling security," Proceedings of the Agent Oriented Methodologies Workshop (OOPSLA 2002), 2002.

[18] H. Mouratidis, P. Giorgini, and G. Manson, "Integrating security and systems engineering: towards the modeling of secure information systems," Proceedings of the 15th Conference on Advance Information Systems (CAISE03), 2004.

[19] B.W. Boehm, "A spiral model of software development and enhancement," IEEE Computer 21 (5) (1988), pp. 61–72.

[20] P. Checkland, Systems Thinking, Systems Practice, Wiley, Chichester,1985.

[21] Tropos Project, Available online: http://www.troposproject.org/, 23/4/2010.

[22] J. McDermott, "Abuse-case-based assurance arguments,' Proceedings of the 17th Annual Computer Security Applications Conference (ACSAC), 2001.

[23] L. Sindre and A. Opdahl, "Eliciting security requirements with misuse cases," Computer Science and Engineering 10 (1) (2005), pp. 34–44.

[24] R.K. Thomas and R.S. Sandhu, "Conceptual foundations for a model of task-based authorizations," Proceedings of the 7th IEEE Computer Security Foundations Workshop, 1994.

[25] R. Strens and J. Dobson, "How responsibility modeling leads to security requirements," Proceedings of the 1992 & 1993 ACM SIGCAS New Security Paradigm Workshop, 1993.

[26] G. Herrmann and G. Pernul, "Towards security semantics in workflow management," Proceedings of the 31st Hawaii International Conference on Systems Sciences, 1998.

[27] A.W. Rohm and G. Pernul, "COPS: a model and infrastructure for secure and fair electronic markets," Decision Support Systems 29 (4) (2000) pp. 434–455.

[28] G. Pernul, "Security constraint processing during multilevel secure database design," Proceedings of the 8th Annual Computer Security Applications Conference, 1992.

[29] E. Ellmer, G. Pernul, and G. Kappel, "Object-oriented modeling of security semantics," Proceedings of the 11th Annual Computer Society Applications Conference (ACSAC'95), 1995.

[30] G. Pernul, A.M. Tjoa, and W. Winiwarter, "Modeling data secrecy and integrity," Data & Knowledge Engineering 26 (1998) 291–308.

[31] T. Priebe and G. Pernul, "Towards OLAP Security Design – Survey and Research Issues," DOLAP, 2000, pp. 33–40.

[32] W. Hutchinson and M. Warren, "Using the viable systems model to develop an understanding information system security threats to an organization," Proceedings of the 1st Australian Information Security Management Workshop, 2000.

[33] M. Karyda, S. Kokolakis, and E. Kiountouzis, "Redefining information systems security: viable information systems," Proceedings of the IFIP TC11 16th International Conference on Information Security (IFIP/SEC'01), June 11–13, Paris, France, 2001.

[34] J. McLean, "Security Models and Information Flow," Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy, pp. 177-186. IEEE Press. May 1990.

[35] D. Parker, "Toward a New Framework for Information Security," in The Computer Security Handbook, 4th ed., edited by Seymour Bosworth and M. E. Kabay, New York, NY: John Wiley & Sons, 2002, ISBN 0471412589.

[36] S. H. Mirjalili and A. Lenstra, "Security Observance throughout the Life-Cycle of Embedded Systems," ESA 2008, pp. 186-192.

[37] D. Parker, "Computer Security Management," Prentice Hall, 1981, pp. 55-82.

[38] R. H. Courtney Jr., "Security risk assessment in electronic data processing systems," AFIPS National Computer Conference, 1977, pp. 97-104.