2008 IEEE/IFIP International Conference on Embedded and Ubiquitous Computing

A Systematic Framework for Structured Object-Oriented Security Requirements Analysis in Embedded Systems

Sojan Markose, Xiaoqing (Frank) Liu, and Bruce McMillin
Computer Science Department
Missouri University of Science and Technology
1870 Miner Circle
Rolla, MO 65409-0350
fliu@mst.edu

ABSTRACT

The primary goal of this paper is to develop a structured object-oriented security requirements analysis methodology for the elicitation and analysis of security requirements in embedded systems. There are several approaches to elicit, analyze and specify security requirements in embedded systems ranging from formal mathematical models for proof of certain security properties to informal methods that are easily understood. Applicability of formal security models is limited because they are complex and it is time consuming to develop. On the other hand, informal security requirements analysis methods are not integrated with conceptual models in requirements analysis, and although both external and internal threats have been dealt using use cases and misuse cases, they provide no process for analyzing both internal and external threats in a structured manner. This paper discusses a structured object-oriented security requirements analysis methodology for the elicitation and analysis of security requirements in embedded systems. It is capable of identifying hierarchically both external and internal threats posed by both external and internal actors of a system level by level. It is illustrated and validated by security requirements analysis for an advanced embedded power grid control system.

Keywords

Security requirements, use cases, misuse cases, structured object-oriented analysis, security goal, and security requirements.

1. INTRODUCTION

Non functional aspects of an embedded system such as security, safety and reliability should be considered and incorporated in the system along with the functional requirements throughout its development process. Among the nonfunctional requirements, security requirements are of great importance because security failure of an embedded system may incur important political, financial and military loses. However security needs of an embedded system are often considered only after the implementation begins. We need to analyze security needs early in the process and integrate counter-measures to security threats into the system. There is also a need for a method which can identify security requirements at multiple levels of a system to deal with both external and internal threats rather than specifying the security requirements for the system as a whole. The existing methods for security requirements analysis do not address the issue of analyzing both internal and external threats in a structured manner. They only deal with the external mis-users of the system.

A systematic process for a structured object-oriented security requirements analysis and specification is needed because objects represent the assets of a system. A Structured object-oriented methodology can be used to model security requirements. This paper i) studies a few noticeable research on extending use cases to elicit and analyze security requirements and ii) introduces a new methodology where security requirements elicitation and analysis is done in a structured manner using HOOMT[7,8] and misuse cases.

2. RELATED WORK

Security requirements are usually specified to prevent any activities that may pose a threat to either the stakeholders or the system itself. Researchers believe that activities that pose a threat to the system can be effectively described using use cases by extending it to represent misuses and mis-users. These extended use cases are called misuse cases and are helpful in documenting negative scenarios which can be used to improve security by preventing them [3]. Sindre et al. developed a method to elicit security requirements using misuse cases [3]. In this method use cases and misuse cases are specified in a single diagram. In addition to the “include” and “extends” relations between use cases in UML, new relations called “prevents” and “detects” were introduced in this method. Ian Alexander developed a method for security requirements analysis using misuse cases in the context of trade-off analysis [4]. An ‘exception’ is considered to be an undesired event that could cause a system to fail and misuse case analysis is the best way to find possible ‘exceptions’. To analyze a trade-off in this method between the use cases and misuse cases two relationships; “threatens” and “mitigates” were introduced. Two more relations “aggravates” and “conflicts with” are also introduced. John McDermott and Chris Fox proposed an abuse case model to capture and analyze security requirements. They define abuse case as ‘a specification of a type of complete interaction between a system and one or more actors, where the results of the interaction are harmful to the system, one of the actors, or one of the stake holders in the system’ [5]. The authors also pointed out that because it is not sure where flaws will occur, an abuse case describes a family of undesirable interactions [6]. The above misuse case based approaches have provided a solid guideline for security requirements analysis. However, several important issues remain to be resolved, such as the integration of analysis of misuse cases and negative scenarios with conceptual models which describes assets of a system, such as object models; and the analysis of both external and internal threats in a structured process.

3. OUR APPROACH

The primary objective of this research is to develop a systematic framework for structured object-oriented security requirements analysis to elicit and analyze security requirements in an embedded system. The proposed framework is expected to improve the existing methods introduced in the previous section by developing a structured object-oriented security requirements analysis process to identify both internal and external threats level

978-0-7695-3492-3/08 $25.00 © 2008 IEEE
DOI 10.1109/EUC.2008.92

by level. The system structure is modeled by a high-order object model based on a High-order Object-oriented Modeling Technique (HOOMT) [7, 8]. The HOOMT provides a structured object-oriented design methodology which is based on hierarchical model development. The term ‘decompose’ in this research represents a process that reveals the sub-components at a lower level of an object, use case, misuse case, security use case or a security requirement.

Figure 1 illustrates a structured object-oriented security requirements process. In the first step a context object diagram is developed which shows the interactions between the high-order system object and the external objects. In the second step, at each level of decomposition, use cases are specified to identify the main functionalities of an object. In the third step, misuse cases and mis-users which can cause harm to the functionalities that have been represented by use cases are identified. At all levels, mis-users can be either external actors or internal actors. There are various relations through which use cases and misuse cases can be related. In this paper we elicit and analyze the security requirements of a FACTS Power System. In order to present an example and describe the elicitation and analysis process, we are considering three actions namely ‘disable’, ‘distort’, and ‘disclose’. The use cases and misuse cases are related using these three actions i.e. ‘disable’, ‘distort’, and ‘disclose’ respectively. The ‘disable’ relates a use case and a misuse case where the misuse case completely disables a functionality represented by a use case. The ‘distort’ relates a use case to a misuse case where the misuse case distorts the functionality represented by the use case. Finally, the ‘disclose’ relates a use case and a misuse case where the misuse case discloses important information about the entities that get benefited by a functionality of the system represented by a use case. In the fourth step security use cases are derived which act as countermeasures to the misuse cases. These security use cases are related to the misuses cases using ‘prevents’ and ‘mitigates’ relations. The ‘prevents’ relates a security use case and a misuse case where a security use case completely stops a negative scenario represented by a misuse case from happening. The ‘mitigates’ relates a security use case and a misuse case where a security use cases reduces the threat of a misuse case. Finally at step five security requirements are derived from security use cases. The security use cases and security requirements are related using a relation called ‘satisfy’. The ‘satisfy’ relates a security use case and a security requirement where the security use case satisfies the derived security requirement. At each subsequent level of decomposition all the above five steps are repeated until a stage is reached where all the objects are primitive and further decomposition is impossible or unnecessary. At the end of this structured object oriented security requirements process, we obtain a set of security requirements which are capable of securing the target system at each level of abstraction.

A detailed process for the hierarchical development and analysis of misuse cases based on high-order objects models and use case diagrams is shown in Figure 2. At the top level a context object diagram is developed which shows the interactions between the high order system object and its external objects. Starting from the top level, security use case diagrams are developed corresponding to the misuse case diagrams. These security use cases are related to the misuse cases using ‘mitigates’ and ‘prevents’ relations. At each hierarchical level security requirements are derived from security use cases.

The decomposition process continues until a stage is reached where the objects are primitive and corresponding use cases and misuse cases are fully explored, Figure 3 represents a detailed process for the hierarchical development and analysis of security requirements based on misuse case diagrams and security use case diagrams. These security use cases are related to the

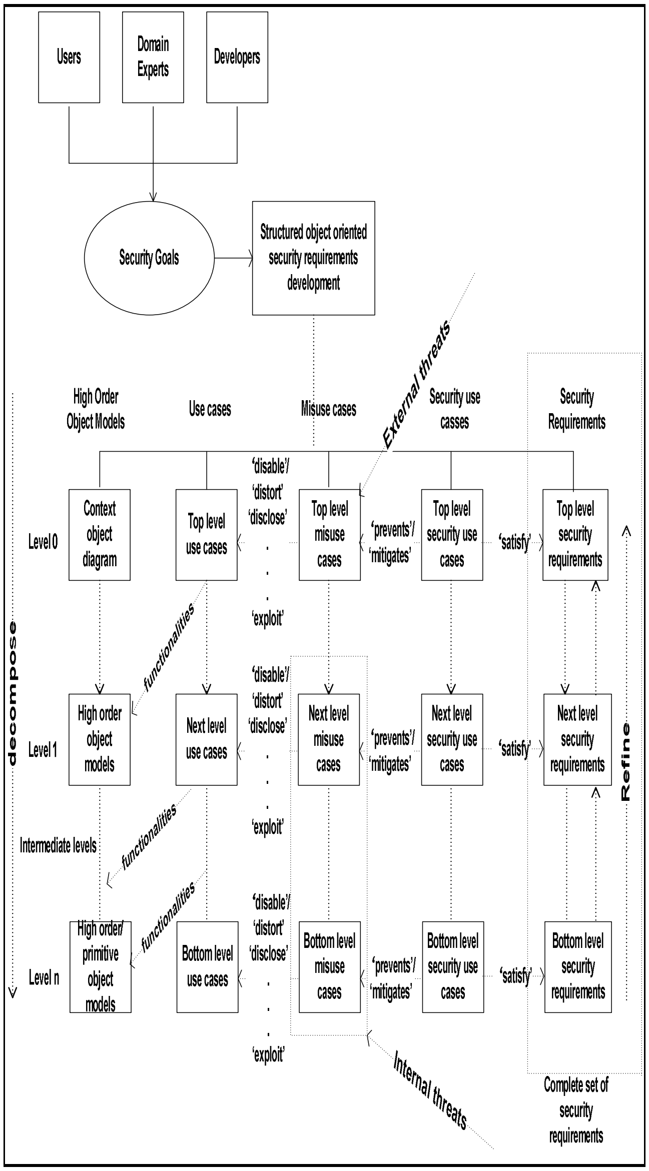


Figure 1: Structures object-oriented security requirements process

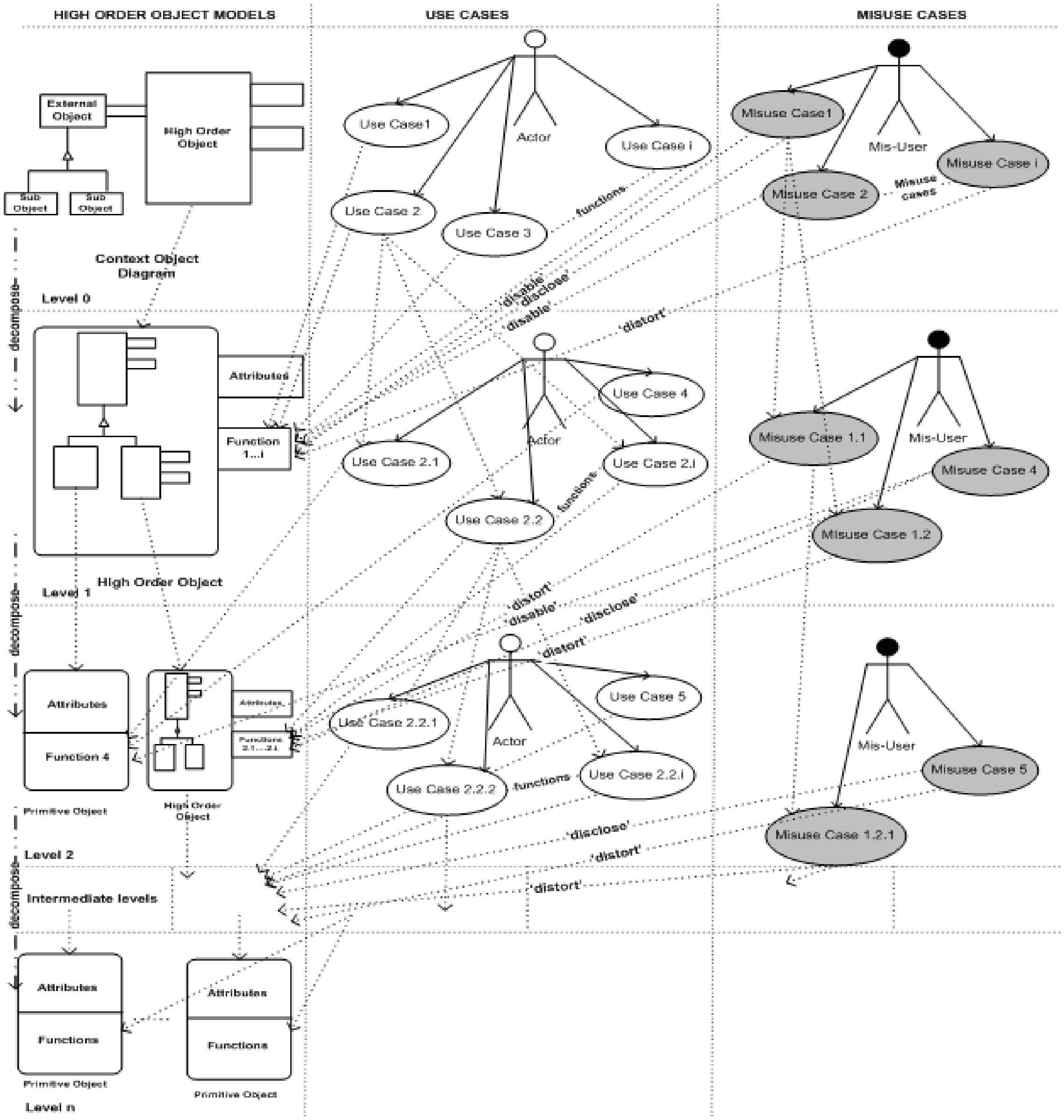


Figure 2: A detailed process for the hierarchical development and analysis of misuse cases

security requirements using the 'satisfies' relation. The decomposition process is continued until a stage is reached where each misuse case at all levels is prevented or mitigated using security use cases and finally corresponding security requirements are derived. As an addition to the proposed systematic framework, a 'misuse sequence diagram' is introduced that will help to better explain a misuse case scenario. Figure 4 represents a misuse sequence diagram, which shows the sequence of message passing that takes place in a misuse case scenario. It describes a misuse case initiated by a mis-user where the mis-user blocks some of the data sent by one of the objects. Object 1 sends data to Object 2 which in turn sends it to Object 3. The Object 3 sends back the acknowledgement to Object 1 through Object 2. The mis-user object, represented by a shaded rectangle, blocks some of the data from reaching the destination. Misuse sequence diagrams can be helpful in deriving security requirements because they help to better explain misuse cases.

4. APPLICATION EXAMPLES

This section is used to describe the application examples on which the proposed systematic framework is implemented. It is based on development of a simulated FACTS power system, where a number of FACTS devices are incorporated into a power grid network to act as a distributed, fault-tolerant, and real-time constrained embedded control system.

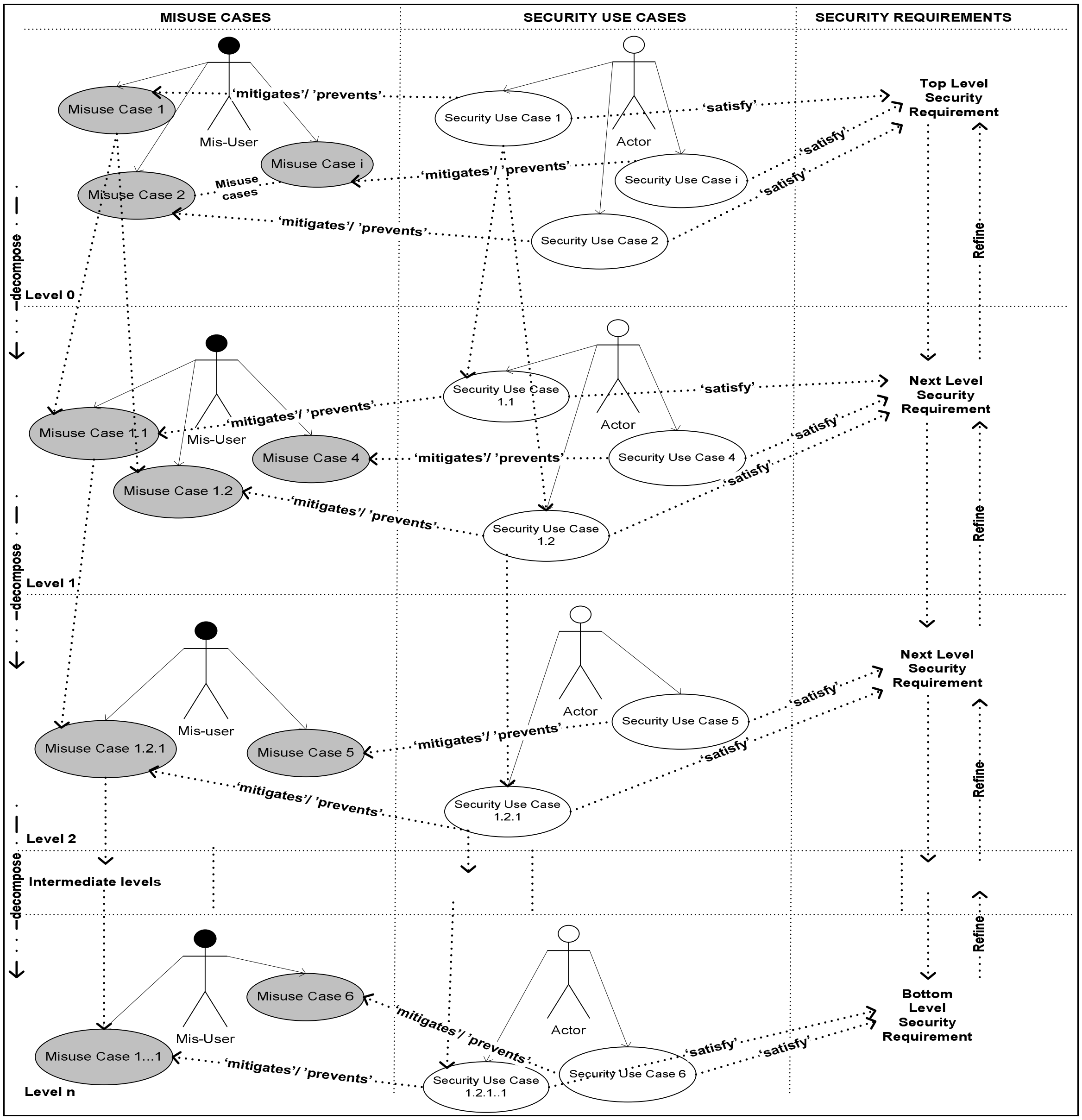


Figure 3: A detailed process for the hierarchical development and analysis of security requirements

Control of power networks is a tedious task because of its sheer size. They are vulnerable to contingencies like line failure. A combination of such contingencies may lead to a cascading power failure. The family of "Flexible Alternating Current Transmission System" (FACTS) devices shows promise for use as network-embedded controllers [1, 2]. There is an ongoing research to incorporate a number of FACTS devices into a power grid network to act as a distributed, fault-tolerant, and real-time constrained control system. The 'Structured Object-Oriented Security Requirements Process' is implemented on the FACTS power system to identify the misuse cases and mis-users that can cause harm to the power system and finally derive security requirements. The high-order object model for the target system was previously developed as part of a research on object-oriented co-analysis/co-design of the FACTS power system [8]. Figure 5 shows the context object diagram for the FACTS power system. The context object diagram shows the interactions between the high-order system object which in this target system is the 'FACTS Power System.' and external objects 'Contingency' and 'Service Provider'. The context object diagram is then further decomposed into a set of components and their relationships in the following high-order object diagrams. The 'FACTS Power System' system object is decomposed into its high-order objects such as 'Facts device', 'Placement' and 'Power transmission system' in Figure 6. Figure 7 represents the 'Use case-Misuse case' diagram of the corresponding level of hierarchy respectively. Once the use cases for each object namely 'FACTS device', 'Placement' and 'Power transmission system' are developed, we obtain the security requirements also.

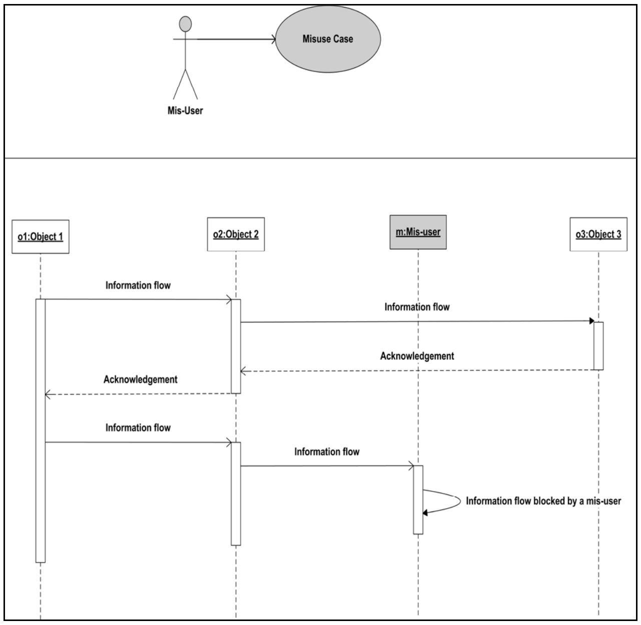


Figure 4: Misuse Sequence Diagram

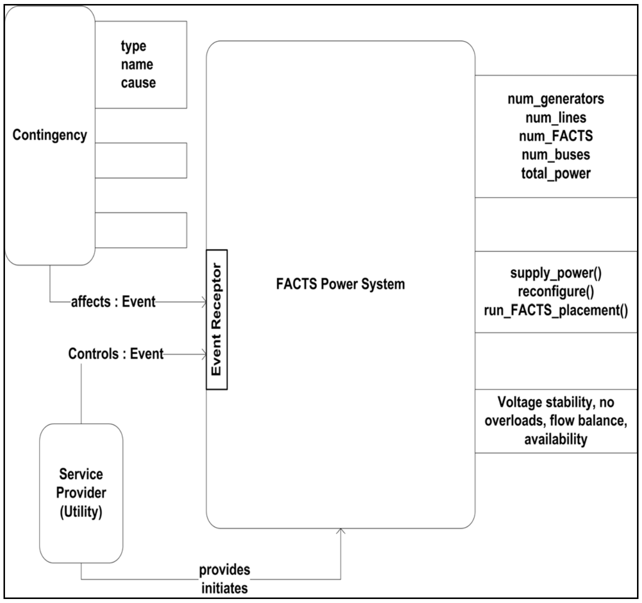


Figure 5: Context object diagram for Facts Power System

At the 'FACTS Power System' object level four significant misuse cases are identified such as 'Physical destruction of FACTS device', Physical destruction of power lines', 'Change FACTS device settings' and 'Interpretation of exchanged messages'. These misuse cases are initiated by the misusers 'Contingency' and 'Intentional'. Based on the identified misuse cases, counter-measures are derived which are represented as security use cases in Figure 7. Finally security requirements are developed from the security use cases. Figure 8 shows the top level security requirements for the target system.

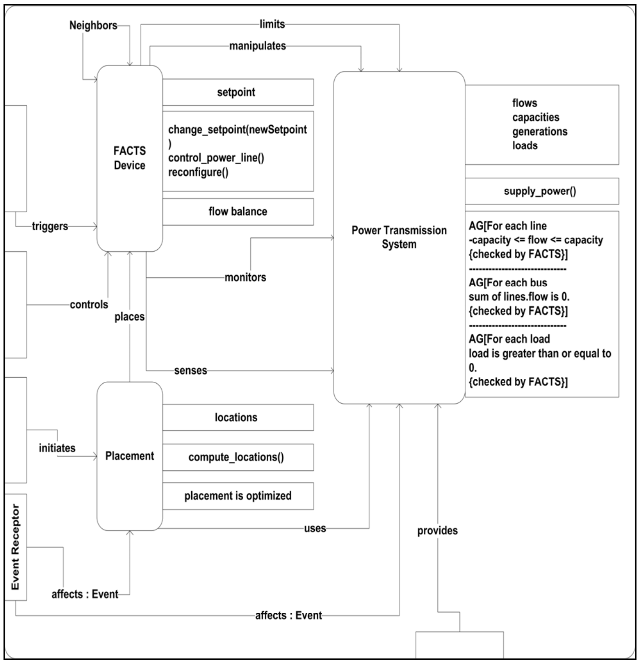


Figure 6: Decomposition of 'Facts Power System' object

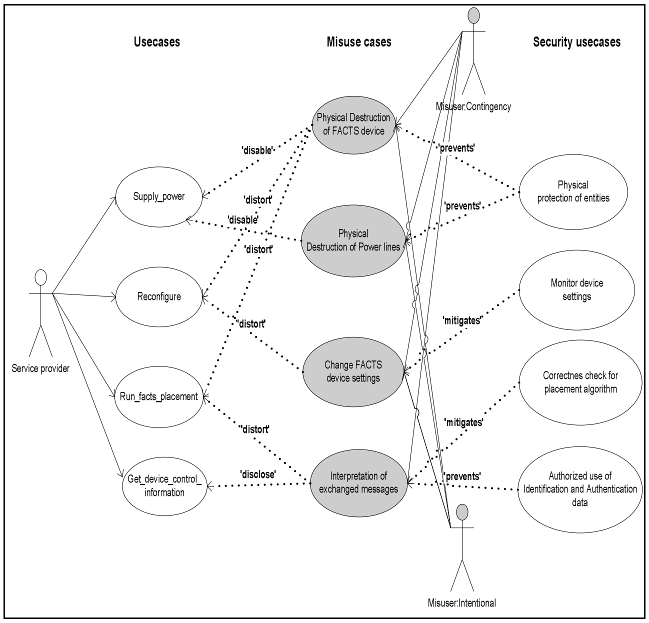
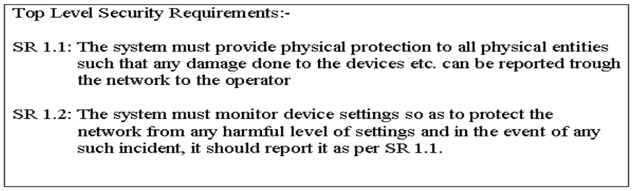

Usecases
Misuse cases
Security usecases
Misuser:Contingency
Physical Destruction of FACTS device
'disable'
'prevents'
Supply_power
'distort'
'disable'
'prevents'
Physical protection of entities
Physical Destruction of Power lines
'distort'
Service provider
Reconfigure
'distort'
Monitor device settings
'mitigates'
Change FACTS device settings
Run_facts_placement
Correctnes check for placement algorithm
'mitigates'
'distort'
Get_device_control_ information
'disclose'
Interpretation of exchanged messages
'prevents'
Authorized use of Identification and Authentication data
Misuser:Intentional

Figure 7: Use case-Misuse case diagram for 'Facts Power System' object


Top Level Security Requirements:-

SR 1.1: The system must provide physical protection to all physical entities such that any damage done to the devices etc. can be reported trough the network to the operator

SR 1.2: The system must monitor device settings so as to protect the network from any harmful level of settings and in the event of any such incident, it should report it as per SR 1.1.

Figure 8: Security requirements for 'Facts Power System' object

At the second level 'UPFC Facts device' object is decomposed into following four sub objects, 'DSP board', 'Embedded computer', 'Interface board' and 'UPFC Power electronics'. Figure 9 represents the high-order object model for 'UPFC Facts device object'. It shows the relationship between the four sub objects.

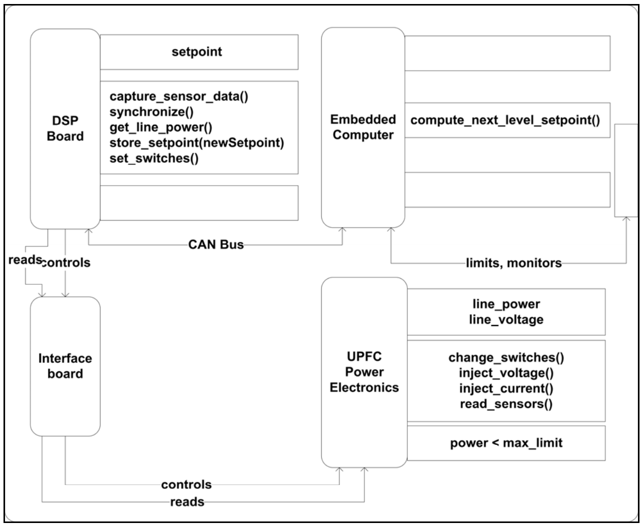


Figure 9: Decomposition of 'UPFC Facts Device' object

The 'UPFC Facts Device' object has use cases, such as 'Control_power_line', Change_set_point','Read _sensor_data' and 'Record_settings'. The misusers 'Contingency' and 'Intentional' initiates three misuse cases such as 'change set point of long term control', 'Change control point in dynamic control', 'Change sensor data in DSP board' and 'Modify Updated Settings'. Counter-measures which are represented as security use cases in Figure 10, are derived from the use cases.

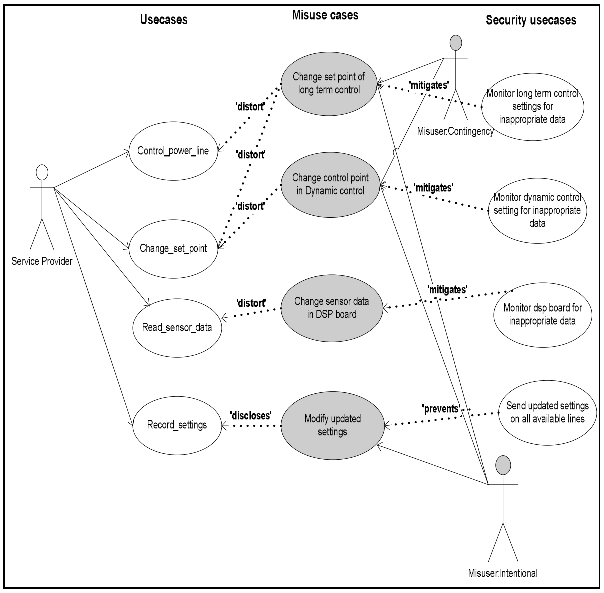
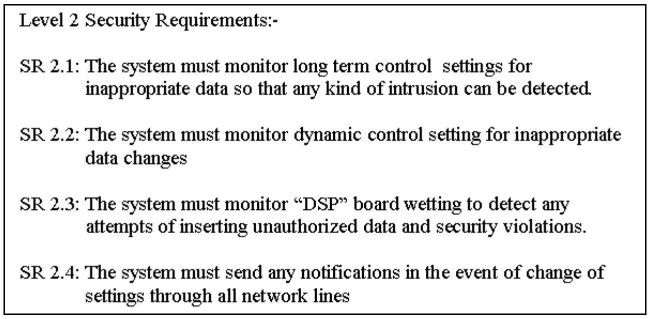

Usecases
Misuse cases
Security usecases
Service Provider
Control_power_line
Change_set_point
Read_sensor_data
Record_settings
'distort'
'distort'
'distort'
'distort'
'discloses'
Change set point of long term control
Change control point in Dynamic control
Change sensor data in DSP board
Modify updated settings
'mitigates'
'mitigates'
'mitigates'
'prevents'
Misuser:Contingency
Monitor long term control settings for inappropriate data
Monitor dynamic control setting for inappropriate data
Monitor dsp board for inappropriate data
Send updated settings on all available lines
Misuser:Intentional

Figure 10: Use case – Misuse case diagram for 'UPFC Facts Device' object

Finally security requirements are developed from the security use cases. Figure 11 shows the security requirements for the 'Facts UPFC Device' object level. At the third level 'Embedded computer' object is decomposed into 'Dynamic control' and 'Long term control' sub objects.


Level 2 Security Requirements:-

SR 2.1: The system must monitor long term control settings for inappropriate data so that any kind of intrusion can be detected.

SR 2.2: The system must monitor dynamic control setting for inappropriate data changes

SR 2.3: The system must monitor "DSP" board wetting to detect any attempts of inserting unauthorized data and security violations.

SR 2.4: The system must send any notifications in the event of change of settings through all network lines

Figure 11: Security requirements for 'UPFC Facts Device' object

Figure 12 represents the high-order object model for the 'Embedded computer' object. The long term control takes sensor data from a power grid, and when necessary it computes new power flows and generations, and then it loads them to compensate for contingencies (such as line outages) in the power grid. The function of the dynamic control is to respond in the short term to fluctuations in the power line attached to the FACTS device, in addition to providing a smooth transition from the current power settings to new set points generated by the long term control. The decomposition also shows the connection of the long term control to the simulation engine, a connection that represents the sending of simulated sensor data to the long term control. The decomposition also shows any changes in generation settings sent from the long term control back to the simulation engine to be applied to the simulation.

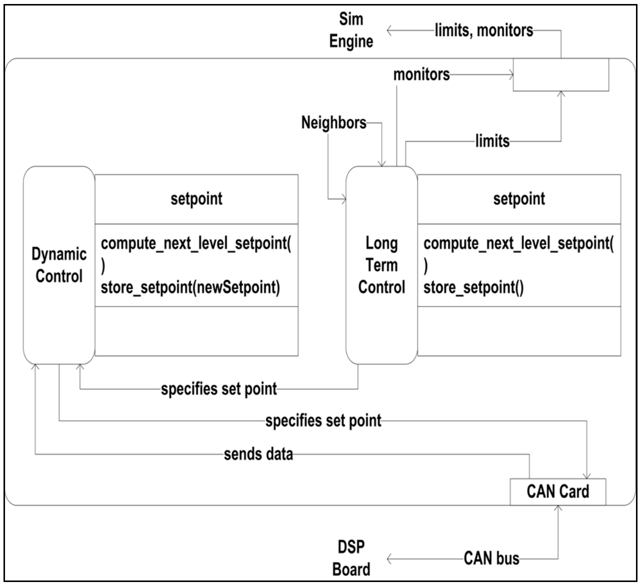


Figure 12: Decomposition of 'Embedded computer' object

One of the major use cases in 'Embedded computer' object is 'compute_next_level_setpoint'. The mis-users 'Contingency' and

'Intentional' initiates three misuse cases such as 'Randomly lose flow messages', 'Alter all flow messages', Randomly invert accept/reject messages', and 'Increase edge flow'. Based on the identified misuse cases, counter-measures are derived which are represented as security use cases in Figure 13. Finally, security requirements are developed from the security use cases. Figure 14 shows the security requirements for the 'Embedded computer' object level.

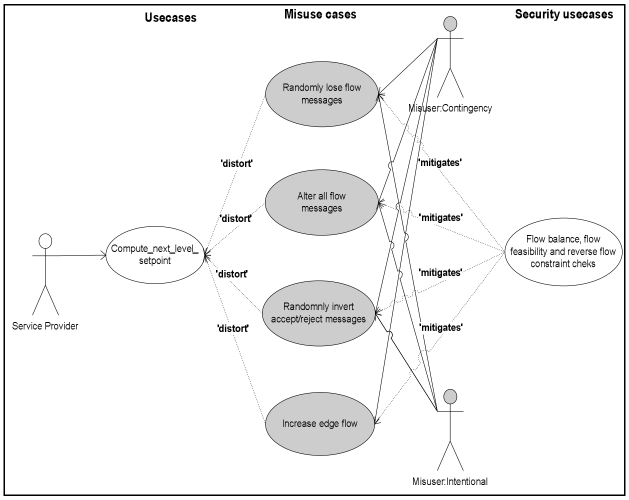
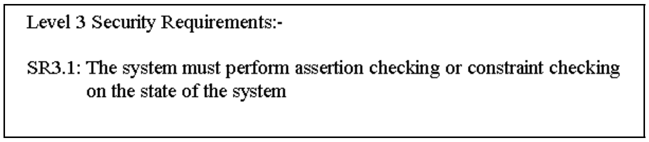


Figure 13: Use case –Misuse case diagram for 'Embedded computer' object



Level 3 Security Requirements:-

SR3.1: The system must perform assertion checking or constraint checking on the state of the system

Figure 14: Security requirements for 'Embedded computer' object

The Figure 15 represents the misuse sequence diagram for the 'Randomly Lose Flow Messages' misuse case.

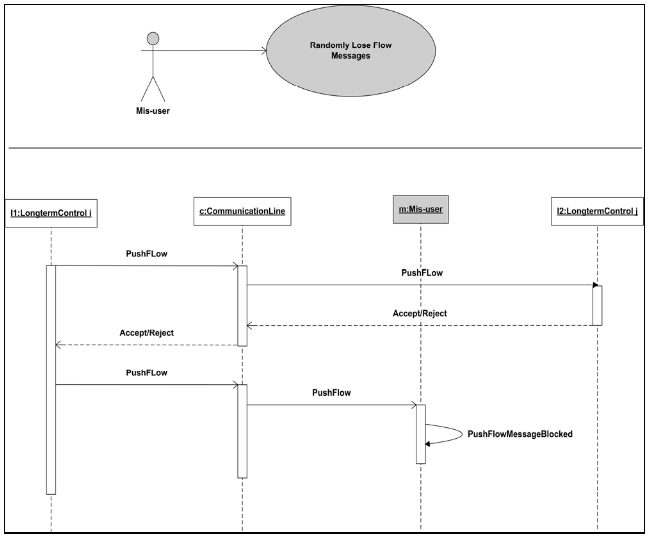


Figure 15: Misuse Sequence Diagram for 'Randomly Lose Flow Messages' Misuse Case

5. CONCLUSION

Security requirements are of great importance these days as many embedded systems are under threat. Although existing methods have provided a useful guideline for security requirements analysis, several important issues remain to be addressed including integration of security requirements analysis methods with conceptual models such as object models of system assets, and analysis of both internal and external threats in a structured manner. This paper successfully addresses these issues by proposing a structured object-oriented security requirements process in which security requirements for the entire system is derived level by level in a structured manner. This methodology is capable of analyzing security requirements by identifying threats posed by both external and internal actors of a system. Object-oriented nature of this methodology helps in identifying the assets of a system which are basically the system objects. To test the applicability, the proposed methodology is successfully demonstrated and applied to a real time FACTS power system. As a result misuse cases at three different levels of hierarchy could be identified. Counter-measures for these misuse cases are identified as security use cases. Finally security requirements are derived based on the security use cases. Security requirements derived at each level of hierarchy when grouped together represents the security requirements specification for the entire system.

6. ACKNOWLEDGEMENTS

This work is supported in part by NSF MRI grant CNS-0420869
Thanks to Vishal Sadana for improving this paper.

7. REFERENCES

[1] Austin Armbruster, Mike Gosnell, Bruce McMillin, Mariesa Crow. "Power Transmission Control Using Distributed Max Flow." Proceedings of the 29th IEEE Annual International Computers Software and Applications Conference. Edinburgh, U.K, June 2005.

[2] B. McMillin, M. L. Crow. "Fault Tolerance and Security for Power Transmission System Configuration with FACTS Devices," Proceedings of the 32rd Annual North American Power Symposium, vol. 1, Waterloo, Ontario, October 2000.

[3] Guttorm Sindre, Andreas L. Opdahl, "Eliciting Security Requirements by Misuse Cases." Proceedings of the 37th International Conference on Technology of Object-oriented languages and systems. Sydney, November 2000.

[4] Ian Alexander, "Misuse Cases: Use cases with Hostile Intent," Software IEEE. January 2003.

[5] John McDermott, Chris Fox, "Using Abuse Case Models for security Requirements Analysis." 15th Annual Computer Security Applications Conference. Arizona. December 1999.

[6] John McDermott, "Abuse-Case-Based Assurance Arguments." 17th Annual Computer Security Applications Conference. New Orleans, December 2001.

[7] X. F. Liu, H. Lin, and L. Dong. "High-Order Object-Oriented Modeling Technique for Structured Object-Oriented Analysis." International Journal of Computer and Information Science (IJCIS), June 2001.

[8] M. Ryan, S. Markose, Y. Cheng, X. F. Liu, B. McMillin. "Structure Object-Oriented Co-analysis/Co-design of Hardware/Software for the FACTS Power System." Proceedings of the 29th IEEE Annual International Computers Software and Applications Conference. Edinburgh, U.K, June 2005.