International Journal on Recent and Innovation Trends in Computing and Communication ISSN: 2321-8169
Volume: 3 Issue: 7 4845 - 4848

Formal Methods for the Verification of Safety Critical Applications using SPIN Model Checker

Neha Chopra
M. Tech*
BMSCE
Sri Mukatsar Sahib, Punjab
chopra.ecb@gmail.com

Er. Lovnish Bansal
Assistant Professor
BMSCE
Sri Mukatsar Sahib, Punjab
lovnish_bansal@rediffmail.com

***Abstract*—** Security over the years has been a major concern for the organizations and companies. With the emergence of smart cards, industry has become more interested in methodologies which are used to establish the correctness and security of the applications developed with the acceptance of the use of smart cards in such domains. This paper provides a general introduction to the state-of-the-art of formal methods for the development of safety-critical systems. The idea is to combine two program verification approaches: the functional verification at the source code level and the verification of high level properties on a formal model built from the program and its specification. One of the important security systems in building security is door access control. The door access control is a physical security that assures the security of a building by limiting access to the building to specific people and by keeping records of such entries. In this paper we employ a model checking method to verify the functional aspects of the smartcard operated door lock system which authenticates each person entering the building. PROMELA model for the proposed system Is presented.

Keywords*—formal methods, formal specification, formal verification, Promela, SPIN*.

*****

I. INTRODUCTION

Over the years, several security measures have been employed to combat the menace of insecurity of lives and property. This is done by preventing unauthorized entrance into buildings through entrance doors using conventional and electronic locks, discrete access code, and biometric methods such as the finger prints, thumb prints, the iris and facial recognition [1]. In this paper, a prototype door security system is designed to allow a privileged user to access a secure keyless door where valid smart card authentication guarantees an entry. This task is accomplished using a well-founded technique of Software Engineering, called Formal Methods. In software engineering, formal methods are mathematically based techniques and tools for the synthesis (i.e. development) and analysis of software systems. Formal methods can be applied at various points through the software development cycle. Formal methods can also be used in reverse engineering to model and analysis existing systems. This paper will focus on formal methods for the specification of functional requirements and design, and for validation/verification which are the most common forms of use of formal methods. The use of formal methods is motivated by the expectation that, as in other engineering disciplines, performing appropriate mathematical modeling and analysis can contribute to the correctness of the resulting product. However, it should be noted that the use of formal methods does not miraculously guarantee correctness, but can be used to increase the level of correctness. Using formal methods in the specification and design phases implies not only that more flaws are found, but also that they are found already in these earlier phases rather than in the testing or maintenance phases. This is also important factor as the cost of repairing flaws is much higher in the later phases than in the earlier phases, e.g. the investigation reported in. Formal methods are usually only used in the development of safety, business, and mission critical software where the cost of faults is high.

In this paper we have given the system (smartcard operated door lock model) description in section 2. In section 3 we discussed about the methodology used to specify as well as to verify the model. Section 4 deal gives the experimental and section 5 we have conclusion of the given study.

II. SYSTEM DESCRIPTION

In this paper it is proposed to construct a simple model to represent a building fitted with a card operated access system is given. For this purpose we have defined the processes required to model the door lock, card readers and users. After that we have combined these processes to create a simulation of a building with a single room accessed through a single door with a card operated lock and a card reader on each side.

Explanation of how the location of each user is represented is given in this model. Simulation takes place in this model using SPIN and shown that when authorized users show their card to a reader, the door opens enabling them to pass through, but the door will not open for unauthorized users. Simulation also illustrates how the locations of users get updated as a result of users passing through doors.
A client server kind of architecture is represented for the above stated problem. Figure 1 shows the abstraction of the model. Here the card reader process will be client and there will be a common server.

 ISSN: 2321-8169
4845 - 4848

A global channel will be used by the card readers to communicate with the server. The server will respond to each card reader on a private exclusive channel which will be passed by the card reader to the server at the time of request.

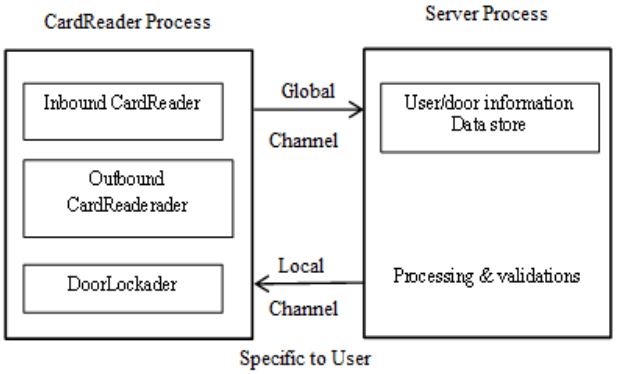

Fig. 1 Design abstraction of doorlock system

Zone 1 & 2 represent the two valid zones in the system. Authorized users can move in and out from zone1 to zone 2.

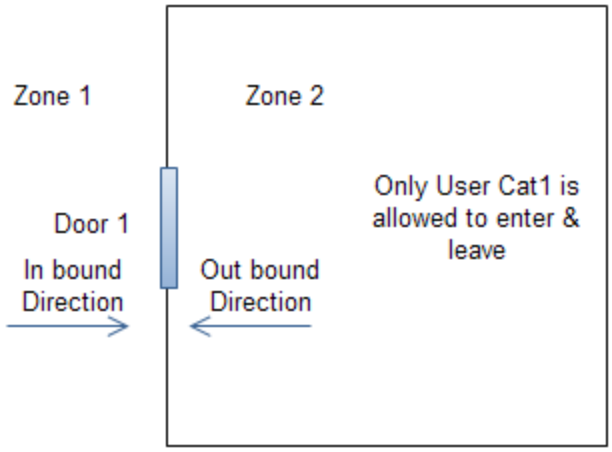

Figure 2: Environment model

We will revise and extend our model with the increased number of doors and zones to represent the building and permissions of inwards and outwards conditions respectively.

III. PROPOSED METHODOLOGY

Our work focuses on verified software implementations and especially on implementations of communication protocols. We extend the PROMELA/SPIN system (one formal description technique) in a way that makes it possible to create compiled implementations of provable correct specifications. The formal description techniques (FDT's) are a technique that can be used by designers of software to ameliorate the quality of their products. Those techniques were initially developed for and are mainly used in the world of telecommunications, but they are getting more and more important for other fields of engineering, like avionics, nuclear power control, medicine, railway control, automotive etc. Figure 3 shows the process of Software Development using Formal Description Techniques [2].

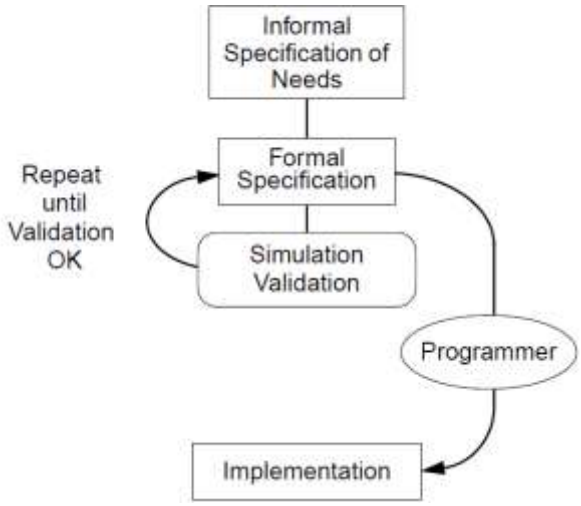

Figure: 3 Software Development using Formal Description Techniques [6]

FORMAL SPECIFICATION

A specification is a description of a product (either to be build or existing). Specifications are used in many different engineering disciplines including software engineering. Formal specifications are, used in the analysis and design phases of the software development cycle to record requirements and design decisions, respectively [2]. They can be used as contracts or communication media between customer and developers, and between developers. To specify our model we have PROMELA modeling language.

Promela is the language for the specification of concurrent systems. Such systems consist of a finite number of separate components, which act independently one from another, and interact through the exchange of messages over message channels [3].

FORMAL VERIFICATION

A classical approach of formal verification consists in building a model of the system in a formal framework, for instance a theorem proven language, and target properties are proved to be satisfied by this model [7]. This approach can be found for instance in industrial domains, when formal methods are used to increase the security level of products. A model of a given sensitive system is usually built from the system requirements specification. Security policies can then be translated into security properties and proved in the same formal framework. In this approach, the implementation is generally developed in parallel with the verification process. Therefore the main problem is to justify the link between the verified model and the implementation. This correctness of the model with respect to the source code is mandatory to claim that the code verifies the target properties. A usual way to strengthen this link is to refine the high level model in lower level models, until a low level model whose link with the code is as straight forward as possible, in terms of data structures and functions [4]. The links between two levels are proved using an abstraction property. There are two major state-of-the-art approaches to formal verification: theorem proving and model checking.

SPIN is a generic model checking tool that supports the design and verification of asynchronous process systems. SPIN verification models are focused on proving the correctness of process interactions, and they attempt to abstract as much as possible from internal sequential computations. Process interactions can be specified in SPIN with rendezvous primitives, with asynchronous message passing through buffered channels, through access to shared variables, or with

any combination of these. In focusing on synchronous control in software systems, rather than synchronous control in hardware systems [6].

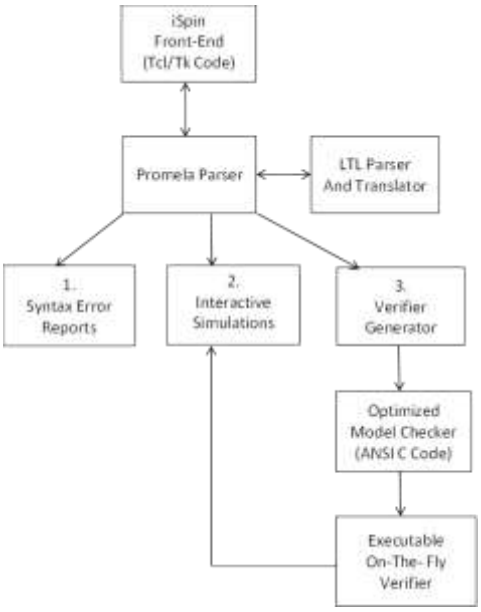

Fig. 2 Basic structure of SPIN [6]

IV. RESULTS OF SPIN SIMULATION

Operations on the card operated door lock system are now examined with SPIN simulator. SPIN is a software tool for analyzing the logical consistency of concurrent and distributed systems. The results obtained with the two defined models clearly demonstrate that partial-order reduction approach can significantly give the results for the state vectors and memory uses when we perform the operations to detect authorized as well as for unauthorized users entering in a building either inbound or outbound directions.

SPIN PARAMETERS USED FOR SIMULATION

S.No.	Name of parameter	value
1.	Safety	
	(i) Invalid endstate (Deadlock)	+
	(ii) Assertion Violation	+
2.	Storage Mode	Exhaustive
3.	Search Mode	Depth First Search(Partial Order Reduction)
4.	Cycle checkes	-
5.	Never Claim	-

SIMULATION RESULTS

Table 1 shows the results of state vector for both the models. SPIN checks for the state of the process by reading the bytes in sate vector global variable. For task 1 it consumed 60 bytes and for task 2 it took 120 bytes since task 2 scales the model with 4 doors at a time.

Name of the Property	Task 1	Task 2
State Vector	60 byte	120 byte
Depth Reached	21	76
State Stored	11	46
States Matched	1	45
Total Transactions (Stored + matched)	12	91
Atomic Steps	6	52
Hash Conflicts	0(Resolved)	0(Resolved)

Table 2 shows the results of memory usage for both the models.

Name of the Property	Task 1 (in MB)	Task 2 (in MB)
equivalent memory usage for states (stored*(State-vector + overhead))	0.001	0.006
Memory used for hash table	0.125	0.125
Actual memory usage for states	0.287	0.287
memory used for DFS stack	0.343	0.343
total actual memory usage	0.758	0.761

The result shown above states that the model derived using promela is successfully implemented through spin simulator. Spin converted the whole promela code into c and performed various security properties on that code and gives the best possible values for the permission of accessing the door by an authorized user.

VI. CONCLUSION

With so much usage of smart card applications the security parameters of these assets becomes a mandatory step. To build a system with very high level specifications written in relational calculus or algebraic notations construct as well as to analyze and to prove correctness properties in those type of systems developed is useful but is still expensive, as it requires experts. Moreover, a formal link between the models and the actual system implementation is lacking. To accomplish the purpose we have used the well founded technique of software engineering known as formal methods. In these methods we performed two basic operations called formal specification and then verification of those specifications using SPIN model checker. The main focus of the developers is then to build tools generating secure code from verified high level models. The method which we have proposed here allows providing a formal verification at source code level.

In this paper we have introduced a model checking approach to verify a smart card operated door lock system. Firstly we analyzed the general specifications of the system including authentication protocol, the processes that can be the participants of the model and the message floe between those processes. Then we proposed the particular specifications for the model and coded them into PROMELA code and then did some experiments which can prove that the model specified by me can simulate actual authentication process. We have provided the verification results.

FUTURE WORK

Our approach can be easily extended to verify the system using some other verification tools and then compare the results of both verification methods. Also some comparison can be made between Model Checking approach and Theoram proving Approach of Verification.

REFERENCES

[1] Oke, A.O., O.M. Olaniyi, O.T. Arulogun, and O.M. Olaniyan (2009): “Development of a Microcontroller - Controlled Security Door System”. Pacific Journal of Science and Technology. 10(2):398-403.

[2] Anne E. Haxthausen and Jan Peleska. Formal Development and Verification of a Distributed Railway Control System. IEEE Trans- action on Software Engineering, 26(8):687–701, 2000..

[3] G.,]. Holzmann. Design and Validation of Protocols. Prentice-Hall, 1990.

[4] Dines Bjørner. New Results and Current Trends in Formal Techniques for the Development of Software for Transportation Systems. In Proceedings of the Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'2003), Budapest/Hungary. L'Harmattan Hongrie, May 15-16 2003.

[5] J. Andronick, B. Chetali, and O. Ly. Using Coq to Verify Java ard Applet Isolation Properties. In International Conference on Theorem Proving in Higher Order Logics (TPHOLs'03), volume 2758 of LNCS, pages 335–351. Springer-Verlag, September 2003.

[6] G.J. Holzmann, "State Compression in SPIN: Recursive Indexing and Compression Training Runs," Proc. Third SPIN Workshop, Twente Univ., R. Langerak, ed., The Netherlands, Apr. 1997.

[7] S. Gerhart, D. Craigen, and T. Ralston. Observations on industrial practice using formal methods. In Proceedings of the 15th Inter- national Conference on Software Engineering, pages 24–34. IEEE Computer Society Press, April 1993.

[8] Anne E. Haxthausen, Marie Le Bliguet, and Andreas A. Kjær. Modelling and Verification of Relay Interlocking Systems. In Christine Choppy and Oleg Sokolsky, editors, 15th Monterey Workshop: Foundations of Computer Software, Future Trends and Techniques for development, number 6028 in Lecture Notes in Computer Science. Springer, 2010. Invited paper.

[9] Anne E. Haxthausen and Jan Peleska. Formal Development and Verification of a Distributed Railway Control System. IEEE Trans action on Software Engineering, 26(8):687–701, 2000.