JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
30th April 2017, Volume 2, Issue 1, JSEIS, CAOMEI Copyright © 2016-2017
www.jseis.org

Validation of applied security design patterns using test templates in secure aware sdlc

[1] E.R.Aruna, [2] A. Rama Mohan Reddy, [3] K.V. N. Sunitha
[1]Department of IT, Vardhaman College of Engineering
[2]SVU College of Engineering, Tirupathi, India
[3]BVRIT for Women, Hyderabad, India
Email: [1]ayam.ac2003@gmail.com, [2]ramamovansvu@yahoo.com, [3]k.v.n.sunitha@gmail.com

ABSTRACT

Security patterns are security knowledge encapsulated tools, they have significant contributions for supporting the software developers as all the software developers need not to be a security specialists. If the applied patterns are inappropriate, this will create vulnerabilities in the product implementation. Here we proposed a method for validation of applied security design patterns in the implementation phase of software development life cycle (SDLC). In this paper, we are verifying the security pattern by creating reusable test case. As a case study, we are validating the applied security patterns for Patient Monitoring System (PMS) application. We believe, the applied security patterns structure is verified in implementation phase. With this we achieve, the security patterns are successfully structured and verified in design and implementation phase. Thus, we can rename SDLC as secure aware SDLC.

Keywords: secure software; design patterns; software development; patient monitoring system;

1. INTRODUCTION

Currently, resolving the security critical issues are vital because most of the e-services are provided by public and private clouds. Preventing the security issues are mandatory to avoid the big loss due to security threats and vulnerabilities [1]. Security must be considered in every aspect of software development process [2]. Fernandez et al. [3] proposed a methodology for integrating security patterns into each one of the software development stages. All the software developers are not security specialists to deal with the security issues that arise during the product development process.

As per the latest research, there are 400 security patterns, these security patterns are reusable, based on the existing classification we can select the pattern based on the domain specific or life cycle phases. Chris Steel et al designed 23 core security patterns for J2EE applications such as Web Tier(9), Business Tier(7), Web Service Tier(3) and Identity Tier(4) patterns[4].Schumacher et al proposed 25 design-level security patterns [5]. More patterns and pattern catalogs were developed for different types of security objectives, threats, vulnerabilities [6 - 13]. Security patterns alone are not enough for supporting the development lifecycle, since they do not have the systematic procedures for an application throughout the complete software lifecycle. A security design pattern is a reusable security knowledge encapsulated tool that can be frequently applied to recurring security problems. During the requirement phase, for the identified security required asset, we must design that asset along with the security patterns. Here the security pattern is security deriving tool to security properties for the identified security critical asset. Security design pattern template consists of Problem, Forces, Solution (structure and strategies), Consequences, security factors and risks, reality checks and related patterns [14].

A pattern can be frequently used as a structure in design phase, behavior or active process in implementation phase. Due to enormous number of security patterns, there are more number of different classifications. Due to abstract description of security patterns, to pick and use the pattern is burdensome. If the patterns are appropriately selected or applied during design phase, consequently this lead to vulnerabilities in the development. Thus, that we must validate the applied patterns are appropriate or not, whether this will exhibit the correct output for the input during the implementation since all the developers are not security specialists.

Many researchers proposed to represent the security design patterns representation in the model during Architecture phase [30]. This will give structure of the design pattern. Due to abstract descriptions of patterns, this is a burden for developers to know the behavior of the pattern during the implementation phase. Due to poor coupling between representation and implementation of security patterns developer cannot be able to predict the impact of pattern on software product.so that we must verify the applied pattern in the model in implementation phase to make, pattern support to software developers.

This paper is organized as related work in Section II, implementation process of our proposed method as a test case in Section III, case study in Section IV, conclusion and future work in Section V.

2. RELATED WORK

Several research practitioners focused on the verification of security patterns. Abramov et al. (2009) proposed a novel approach that utilizes security patterns for enforcing security over database applications design and injecting security constraints to the database. But this approach lacks in expressiveness of security constraints for low level elements such as attributes. The research study in [15] proposed stereotype. Peng et al. (2008) presented a formal verification method to analyze the behavioral correctness of a design pattern implementation. Their method proposed the partial order relationship between the sequence diagram of a general design pattern and that of its implementation. However, this method does not verify the structural behaviour of the implementation. Therefore, there is a need to develop an approach to automatically and fully validate the implementation of patterns [16].

Dongs et al. (2010) proposed a method to verify the compositions of security pattern using model checking. This approach formally defines the behavioral aspect of Security patterns but it does not conduct analysis at the implementations level even though it is verified at model level [17]. Hamid et al. (2012) proposed a method to validate the applications of patterns at the model level by UML and OCL but this is also not applicable implementation level analysis [18].

Jan Jurjens (2005) proposed UMLsec tool in the form of a UML profile using standard UML extension mechanisms. Stereotypes are used to formulate the security requirements, while constraints are used to verify whether the security requirements hold during a specific type of attack [19]. Lodderstedt et al. (2002) proposed secureUML that focuses on modeling access control policies and how these policies can be integrated into a model-driven software development process. It is based on RBAC model and uses RBAC as a meta-model to specify and enforce security. RBAC lacks support to express access control conditions that refer to the state of a system, such as the state of a protected resource. In addressing this limitation, secureUML introduces the concept of authorization constraints. Authorization constraints are preconditions for granting access to an operation [20]. Both secureUML and UMLsec techniques are supporting modeling security-related concerns in UML models, they cannot be directly used to check the correctness of security pattern applications at the implementation level.

Masatoshi et al. (2016) proposed a TESEM tool, that is used to create a reusable test case template, which is derived from in the consequence of Test Driven Development model (TDD), Object Constraint Language (OCL), Aspect Oriented Programming (crosscutting concerns such as pointcut and advice pair), Junit tool to test the test case. Using this method, a supporting reusable test template is created, which is useful for the developers as a concrete test template for testing the applied security pattern in the implementation phase [21]. But, in this method the designer/researcher must have knowledge of OCL to verify the structure of the design patterns.

3. IMPLEMENTATION PROCESS TO CREATE A TEST TEMPLATE

To implement our method, we used decision table testing, Aspect-Oriented Programming, Junit testing Tool. Our proposed method adopts decision tables to describe the specifications of target security design patterns [22], most of the decision table testing is used for web applications [23]. Aspect-oriented Programming is used to improve the modularity of the software product in terms of crosscutting concerns [24, 25]. JUnit tool for testing Application, once the template is created, then is tested using the JUnit tool [26, 27].

In our method, we created the reusable test template as follows:

1. Select the appropriate security pattern from the available catalogs, we are suggesting to select the pattern based on the annotations of Koen’s proposed classification [28, 29]. During requirements phase, select the pattern for each type of security concern.
2. Represent the selected patterns in the model during architecture phase.
3. Verify the structure of the pattern and analyze the pattern behavior with respect to security required assets.
4. Create a decision table for every security concern provided by pattern, based on true positive and false negative values that is conditions and actions.
5. Use aspect oriented programming, for each condition to analyze the internal processing of the pattern without omitting all the possible escape artifacts based on decision table.
6. Create a test case based on the behavior of the pattern with aspect programming crosscutting concerns.
7. Run the test case with all the possible cases using JUnit tool, fix the bugs.
8. Revalidate the tool by retesting, if the process is successful stop the process.
9. If the test is failure, then repeat the steps from 4 to 7.

JOURNAL OF SOFTWARE ENGINEERING & INTELLIGENT SYSTEMS
ISSN 2518-8739
30th April 2017, Volume 2, Issue 1, JSEIS, CAOMEI Copyright © 2016-2017
www.jseis.org

4. CASE STUDY

We are implementing the above proposed method to electronic health domain application named as Patient Monitoring System [30]. In this application, a doctor is monitoring the patient remotely using the sensors. A wearable unit of the patient is continuously measuring the patient condition such as a pulse rate, blood pressure. These readings are collected and stored in the PMS device (smart phone). The stored PMS database, analyzes and alerts the doctor about the patient health condition if it comes to abnormal condition. When it analyzes and indicates as an emergency, it notifies the emergency services. The same system is accessible by the doctor, patient care taker as well as the patient relatives. They can review the patient's status.

The same system is also connected to the hospital information system (HIS) [30]. In this application, in order to access the patient information by the doctor, the doctor has to authenticate himself by providing the user ID and Password.

Step 1, 2: In this context, we are using the Password Design and Use pattern, Prevent SQL Injection Patterns. The Password Design and Use pattern describes the best security practice to design, create, manage, and use password components [5, 30].

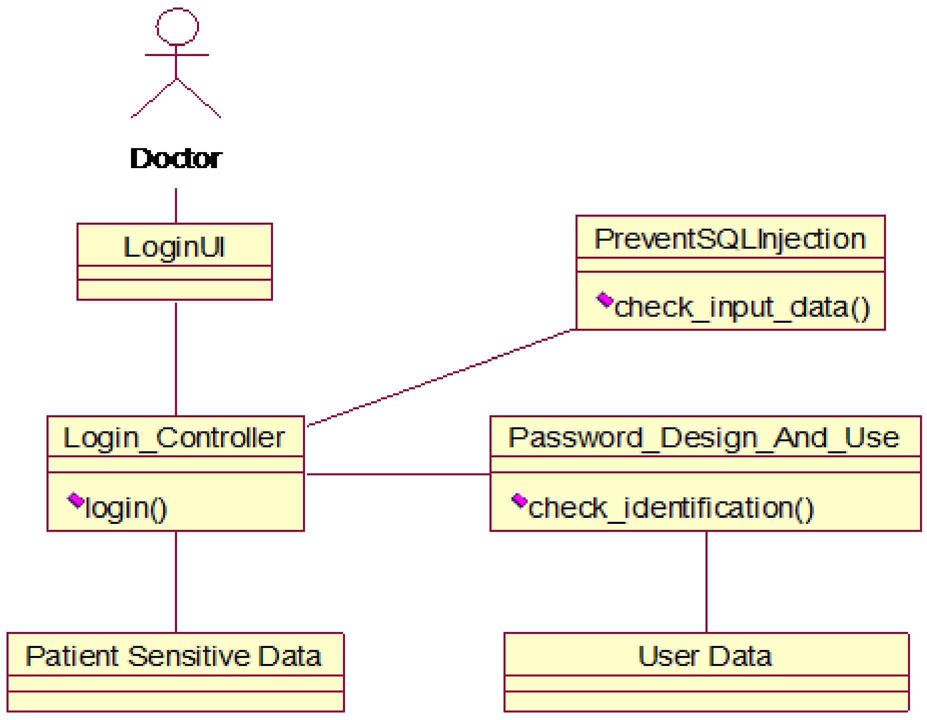


Figure. 1 Login portion of the doctor to access patient data.

Step 3, 4: in this step, we are analyzing the pattern behavior based on the decision table as shown in Table 1.

Table. 1 Decision Table

		1	2	3	4
Conditions	Given ID agrees with User Data	Y	Y	N	N
	Given PW agrees with User Data	Y	N	Y	N
Actions	Considered as Doctor	√			
	Can Access Patient sensitive data	√			
	Considered as a Malicious User		√	√	√
	Can't Access Patient sensitive data		√	√	√

Based on the decision table, we are configuring the aspect oriented crosscutting concerns in the next step.

Step 5, 6: creation of aspect template such as crosscutting concerns in terms of pointcut and advices based on the step 3 & 4.

```
Pointcut LoginCheck();
Call(* *.password_design_and_use.check_identification(..));
```

Figure. 2 Pointcut considering for authorization doctor

```
after() returning(Boolean right):
EscapeCheck() { setTemporary("LoginCheck", right);}
```

Figure. 3 Advice to differentiate the two types of users

```
Pointcut AssetAccess();
Call(* *.Subject_Controller.subject_function(..));
```

Figure. 4 Pointcut judging access to patient sensitive data

```
after() returning(Boolean right):
AssetAccess() { setTemporary("AssetAcess", right);}
```

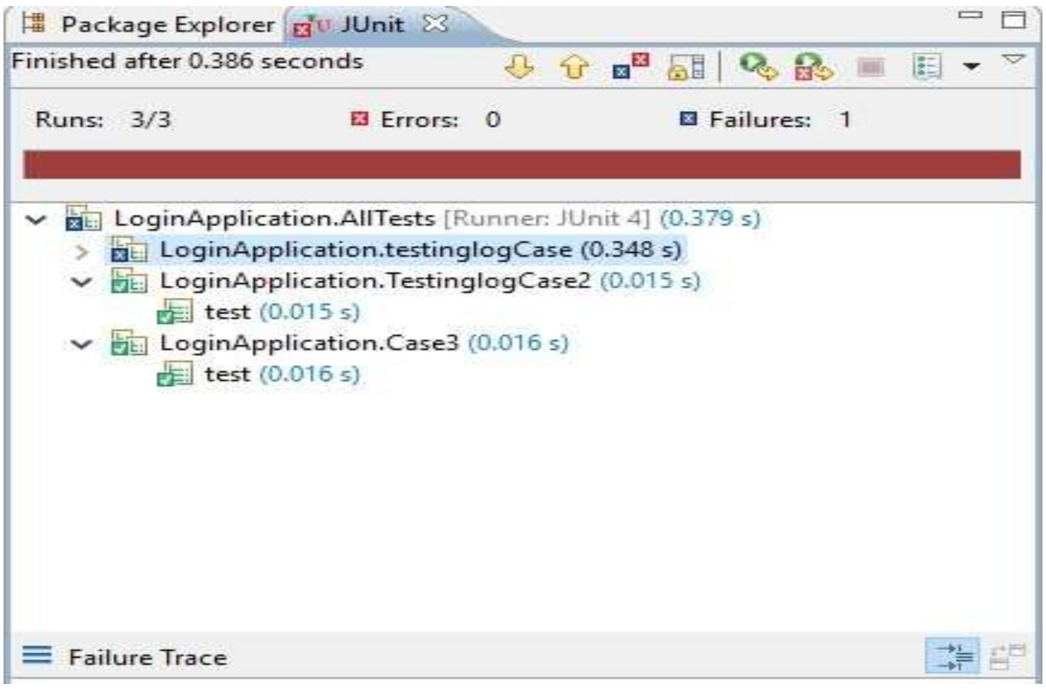
Figure. 5 Advice to allow access patient sensitive data

Figures 2, 3, 4 and 5 show the aspect oriented crosscutting concerns for modular based pointcuts and corresponding advices.

Now we create the test template based on above points without omitting any security aspect.

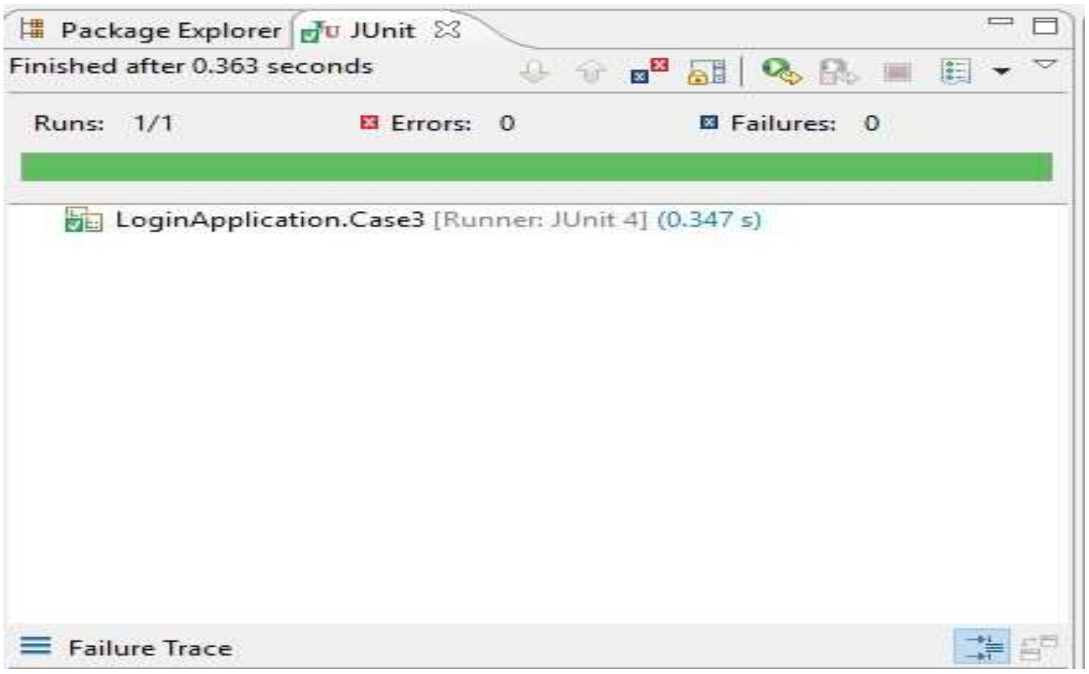
Step: 7, 8, 9:

With JUnit tool run the test case by providing the input, if the results are failure, then retest the template and revalidate until the test case is going to produce the correct outputs for inputs.



```
Finished after 0.386 seconds
Runs: 3/3    Errors: 0    Failures: 1
LoginApplication.AllTests [Runner: JUnit 4] (0.379 s)
  LoginApplication.testinglogCase (0.348 s)
  LoginApplication.TestinglogCase2 (0.015 s)
    test (0.015 s)
  LoginApplication.Case3 (0.016 s)
    test (0.016 s)
```

Figure. 6 Results of the failure test case



Finished after 0.363 seconds

Runs: 1/1 Errors: 0 Failures: 0

LoginApplication.Case3 [Runner: JUnit 4] (0.347 s)

Figure. 7 Results of the success test case

The figures 6 and 7 show the failure and success test case of the Patient Monitoring System Application.

5. CONCLUSION AND FUTURE WORK

Even though the security design patterns are applied properly in the design phase of software design, threats and vulnerabilities may not be mitigated in the implementation phase. Hence, we propose a support method for security design patterns in the SDLC implementation phase. Our proposed method is very flexible to find and fix the implementation defects. Security developer need not to be a security specialist.

In future, we intended to work to apply this method to more security critical applications and patterns. In this, we believe, we have successfully applied patterns in requirement, design and implementation phases and consequently deployment phase. With this we can rename SDLC as secure aware SDLC with our proposed process.

ACKNOWLEDGEMENT

I am immensely happy and acknowledge my profound regards and remain obliged to my Guides throughout and look forward to their blessings in all my future endeavors. I also acknowledge my special thanks to the department of Information Technology, Vardhaman College of Engineering for their earnest support and cooperation enabling me to present my paper.

REFERENCES

1. Yoshioka, N.; Washizaki, H.; Maruyama, K. A Survey on Security Patterns. Prog. Inform. 2008, 5, 35–47.
2. Devanbu, P.T.; Stubblebine, S. Software Engineering for Security: A Roadmap. In Proceedings of the Conference on the Future of Software Engineering, New York, NY, USA, 4–10 June 2000; pp. 227–239.
3. E.B. Fernandez, M.M. Larrondo-Petrie, T. Sorgente, and M. VanHilst, "A Methodology to Develop Secure Systems Using Patterns," Integrating Security and Software Engineering: Advances and Future Vision, H. Mouratidis and P. Giorgini Eds., IDEA Press, Ch. 5, 2006, pp. 107-126.
4. Chris Steel, Ramesh Nagappan, Rau Lai, " Security Patterns for J2EE Applications, Web Services, Identiy Management and service providing".www.coresecuritypattrens.com.
5. Schumacher, M.; Fernandez-Buglioni, E.; Hybertson, D.; Buschmann, F.; Sommerlad, P. Security Patterns:
6. Integrating Security and Systems Engineering; John Wiley & Sons: Chichester, UK, 2006.

7. Kienzle, D. M., Elder, M. C., Tyree, D. and Edwards- Hewitt, J. 2001. Security patterns repository version 1.0. Retrieved 15 December, 2015 from http://www.scrypt.net/~celer/securitypatterns/repository.
8. Romanosky, S. 2003. Enterprise Security Patterns.*Information Systems Security Association Journal*.
9. Blakley, B. and Heath, C. 2004. Security design patterns technical guide - version 1. Open Group (OG). Retrieved 24 November, 2016 from www.opengroup.org/onlinepubs/9299969899/toc.pdf.
10. Trowbridge, D., Cunningham, W., Evans, M., Brader,L. and Slater, P. 2004. Describing the enterprise architectural space. *Microsoft Press.*
11. Hafiz, M. 2006. A collection of privacy design patterns. Proceedings of the 13th Conference on Patterns Language of Programming (PLoP'06).
12. Fernandez, E. B., Yoshioka, N. and Washizaki, H. 2007.Using security patterns to build secure systems. International Workshop on Software Patterns and Quality. Information Processing Society of Japan, (pp. 47–48).
13. Dougherty, C., Sayre, K., Seacord, R.C., Svoboda, D. and Togashi, K. 2009. Secure design patterns. Carnegie Mellon University, Software Engineering Institute, TECHNICAL REPORT CMU/SEI 2009-TR- 010
14. Fernandez-Buglioni, E. 2013. Security Patterns in Practice: Designing Secure Architectures Using Software Patterns. Wiley, ISBN: 978-1-119-99894-5
15. Abramov, J.; Shoval, P.; Sturm, A. Validating and Implementing Security Patterns for Database Applications. In Proceedings of the 3rd International Workshop on Software Patterns and Quality (SPAQu), Orlando, FL, USA, 25 October 2009.
16. T. Peng, J. Dong, and Y. Zhao, "Verifying Behavioral Correctness of Design Pattern Implementation," Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE), CA, USA, July 2008, pp. 454-459.
17. Dong, J.; Peng, T.; Zhao, Y. Automated verification of security pattern compositions. J. Inf. Softw. Technol. 2010, 25, 274–295.
18. Hamid, B.; Percebois, C.; Gouteux, D. Methodology for Integration of Patterns with Validation Purpose. In Proceedings of the European Conference on Pattern Language of Programs (EuroPLoP), Irsee, Germany, 11–15 July 2012; pp. 1–14.
19. Jürjens, J. Secure Systems Development with UML; Springer: Berlin, Germany, 2005.
20. Lodderstedt, T.; Basin, D.A.; Doser, J. SecureUML: A UML-Based Modeling Language for Model-Driven Security. In Proceedings of the 5th International Conference on the Unified Modeling Language (UML), Dresden, Germany, 30 September–4 October 2002; pp. 426–441.
21. Masatoshi Yoshizawa, Hironori Washizaki, Yoshiaki Fukazawa, Takao Okubo, Haruhiko Kaiya and Nobukazu Yoshioka, "Implementation Support of Security Design Patterns Using Test Templates", Information 2016, 7, 34; doi:10.3390/info7020034, ww.mdpi.com/journal/information.
22. Copeland, L. A Practitioner's Guide to Software Test Design; Artech House: London, UK, 2004.
23. Di Lucca, G.A.; Fasolino, A.R.; Faralli, F.; de Carlini, U. Testing Web Applications. In Proceedings of the 18th International Conference on Software Maintenance (ICSM), Montreal, QC, Canada, 3–6 October 2002; pp. 310–319.
24. Endrikat, S.; Hanenberg, S. Is Aspect-Oriented Programming a Rewarding Investment into Future Code Changes? A Socio-technical Study on Development and Maintenance Time. In Proceedings of the International Conference on Program Comprehension, Kingston, ON, Canada, 22–24 June 2011; pp. 51–60.
25. Kiczales, G.; Hilsdale, E.; Hugunin, J.; Kersten, M.; Palm, J.; Griswold, W. An overview of AspectJ. In Proceedings of the Conference on Object-Oriented Programming, Budapest, Hungary, 18–22 June 2001; pp. 327–354.
26. JUnit. Available online: http://junit.org (accessed on 3 June 2016).
27. Tahchiev, P.; Leme, F.; Massol, V.; Gregory, G. JUnit in Action; Manning Publications Co.: Shelter Island, NY, USA, 2010.
28. Koen YSKOUT," Connecting security requirements and software architecture with patterns",Thesis, 2013.
29. Koen Yskout, Riccardo Scandariato, Wouter Joosen, "Does Organizing Security Patterns Focus Architectural Choices?",.
30. MS. E.R. Aruna, DR.A. Rama Mohan Reddy, DR. K.V. N. Sunitha," Design of Patient Monitoring System(PMS) application using Security Design Patterns Architecture Phase of Secure SDLC", IJMTER, DOI:10.21884/IJMTER.2016.3147.WIIHU.

31. Fernandez-Buglioni, E. Security Patterns in Practice: Designing Secure Architectures Using Software Patterns; John Wiley & Sons: Chichester, UK, 2013.

AUTHORS PROFILE

Ms. E.R.Aruna, is working as Associate Professor in Department of Information Technology, Vardhaman College of Engineering. She is pursuing her PhD in Computer Science and Engineering in JNTUH, Hyderabad, India. She received Master if Information Technology in 2008, Sathyabhama University, Chennai, India and received Bachelor of Computer Science and Information Technology in 2004, SITAM, Chittoor, AP, India.

Dr. A. Rama Mohan Reddy, is working as Professor in Computer Science and Engineering, SVU, Tirupathi, India. He has done PhD in CSE, SVU, Tirupathi, India. He has received Master's degree in CSE from NIT, Waranghal, AP, India and Bachelor's degree from JNTU, Annathapur, India.

Dr.K.V.N.Sunitha, is working as Principal and Professor of CSE in BVRIT for Women, Hyderabad, India. She has done her B. Tech (ECE) from Nagarjuna University, M. Tech Computer Science from REC Warangal. She completed her Ph.D from JNTU, Hyderabad in 2006.