Study of Testing of Security Model for an Object Oriented Design (SMOOD)

Sonali P. Kadam[1] Shashank Joshi[2]
[1,2]Department of Computer Engineering
[1,2]Bharati Vidyapeeth Deemed University's College of Engineering, Pune Satara Road, Pune-411043, India

Abstract— Software security involves internal weakness and external attacks. The external threat often breaks a software system by exploiting its internal weakness or vulnerabilities. As a result, they can affect the violation and breakdown of security. Security architecture must be designed to cater the needs of product security goals and sensitive information. To keep the sensitive information confidential, special attention need be given at designing phase. An object oriented approach naturally lends itself to an early assessment and evaluation. To check the security of design author has proposed SMOOD model which is based on the relationship between design properties such as encapsulation, inheritance, complexity, abstraction, cohesion, coupling and polymorphism with security attributes like confidentiality, integrity, and availability (CIA). In this paper, the proposed model is validating using empirical validation reported by Fenton. We have used five open source projects with multiple versions to test the SMOOD model and discussed the results.

Key words: Security, Confidentiality, Integrity, Availability, Validation

I. INTRODUCTION

Object oriented methodologies require significant effort early in the development cycle to identify object, classes, attributes, operations and relationships. Encapsulation, inheritance, and polymorphism needs designer to carefully structure the design and consider the interaction between objects. Therefore, the approach provides the information required to assess the security of the design's classes, structure and relationships before they are committed to an implementation. Literature survey reveals that methods to remove vulnerabilities in developed software are very costly in terms of time, money and efforts [10] [11]. The direct measurement of security is difficult because there is no standard method to measure security factors. For measuring this factor, we have to express in terms of metrics or model. In earlier work author has proposed security model to evaluate the security of object oriented software at design phase of Software Development life cycle(SDLC).Various views as what relationship the design properties are having with security properties has been introduced by different reviews of object oriented development related literature. This is done using the result collection through case studies [9], some existing work done by other researcher [4] [5] [6] [7] [8] and also by surveying the views of industry experts working in related areas. The proposed model can check the security in software architecture level which allows the designer to do the necessary action to deliver the secure software, which is safe by design. The benefits of such work are detecting and removing defects earlier, which in turn reduce development time and cost. The proposed security model for an object oriented design is given in table 1.In this paper; author has attempted to validate the proposed security model. Direct assessment of security is very difficult because it is a multifaceted concept, which varies according to people, environment and application. Only application level security is not sufficient, security has to integrate with software development life cycle to provide system level security. Around 50 percent of the security flaws uncovered during Microsoft security push in 2002 were closely related to design level problems.

The validation of software product metrics means convincingly demonstrating that:

1) The product metric measures what it purports to measure. For example, whether a coupling metric is actually measuring coupling
2) The product metric is associated with some important external metric (such as measures of maintainability or reliability).
3) The product metric is an improvement over existing product metrics. An improvement can mean, for example, that it is easier to collect the metric or that it is a better predictor of faults [12]. According to Fenton [2] [3], validating system in a given environment is the process of establishing the accuracy of the proposed system by empirical means. This can be done by comparing model performance with known data in the given environment.

Security Attribute	Index Computation Equation
Availability	1.5 × polymorphism – 0.25 × Coupling - 0.25 × Complexity
Integrity	0.3 × Abstraction + 1.0 × Encapsulation + 0.7 × Cohesion - 0.20 × Inheritance - 0.6 × Complexity - 0.20 × Coupling
Confidentiality	0.65 × Abstraction + 1.0 × Encapsulation +1.0 × Cohesion - 0.33 × Inheritance - 0.33 × Coupling -0.33 × Polymorphism - 0.66 × complexity

Table 1: Security Model for Object Oriented Design

II. INVESTIGATIVE TECHNIQUE

Three types of investigation were mentioned in empirical validation like experiment, case studies and surveys. A survey is a retrospective study that is done after an event has occurred. Both case studies and formal experiments are, usually, not retrospective. We need to decide in advance that what you want to investigate and then plan how to capture data to support for respective investigation. A case study may be preferable to a formal experiment if the process changes caused by the independent variables are wide ranging, recurring the effects to the measured at a high level and across too many dependent variables to control and measure. In the proposed security model, we observed the changes in security occur by different design properties, so case study approach is more suitable than the survey and formal experiment. In baseline case study technique we can compare our new inspection approach with a general baseline. In this data can be gathered from various projects, regardless of how different one is a project from another one but should have general baseline. In present work, we have collected data from selected open source software implemented with C++ having common baseline. Like pure C++ programming language, the open source project of product is software application, the product is released more than four stable products and the product has long project age and big amount of download.

III. HYPOTHESIS

Usually there are common reasons behind the a new release of existing software like, addition of new feature, changes in existing approach to enhance capabilities, fixing of the defect found in the software. The early versions of the software are unstable and undergo significant rework during initial releases to improve quality after stable released a mature version of the software comes in the market with improved robustness and reliability. So security of this mature version has been observed to be significantly better than its predecessors. Not drastically change but should have minor changes. After released of mature version, the trend is expected to reserve the same for quality and security. It was expected that security characteristics of each version of the software evaluated should match the expected trends from one version to another version. To verify that the computed values of SMOOD that are in a valid range, the security attribute values were compared with expected results.

IV. MAINTAINING CONTROL OVER VARIABLES

Ones we have an explicit hypothesis, it is necessary to decide what variables can affect its truth. In this study we are using five different software on common baseline as mention in section II, nine versions of scum mvm, four versions of keepass, eight version of ivf (interactive visual framework), fourteen version of mockpp, twelve version of g3d. For each version of all software first we calculate design properties with the metric mention in Bansia quality model [1] , from these design properties security attributes confidentiality, integrity and availability are calculated as mention in SMOOD security model. Graph plotted for this software are given below in figure 1 to figure 5.

V. INVESTIGATION OVER HYPOTHESIS

Validation is performed by correlating one measure with another. For the kee Pass software we have four different released, after first version software is showing mature stable results for confidentiality, integrity, availability for next three releases. Confidentiality and integrity is increased and availability remains stable. The expected increase in value of security attributes as compared to its first version to the next one is seen in all versions of keepass software. It is match with expected trends and our hypothesis. For ivf (interactive visual framework) we have eight releases, first three versions showing stable results for CIA , next two versions goes under so many changes so lots of ups and down are shown. After released of the fifth version, software is become mature and giving stable result only slight decrease in of all security attributes in last version. But last few versions confidentiality, integrity and availability have been an increasing as compare to with first few versions For mockpp the software we have fourteen releases. After released of first nine versions software is showing mature result and become stable for confidentiality and integrity with increases in availability.

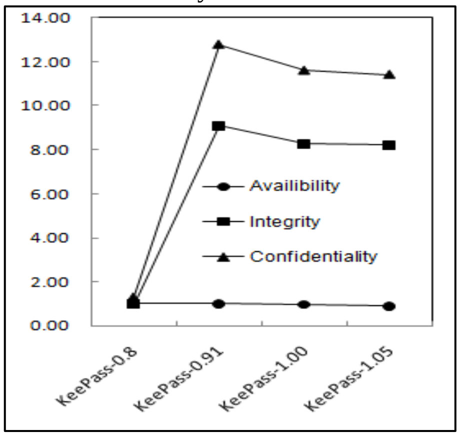

Fig. 1: Plot of Computed Security Attributes Values for Keepass

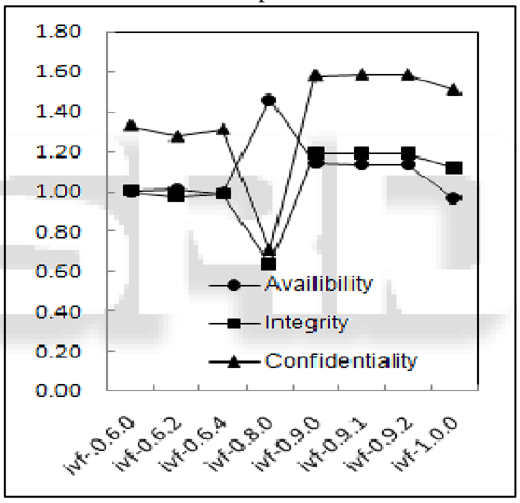

Fig. 2: Plot of Computed Security Attributes Values for IVF Versions

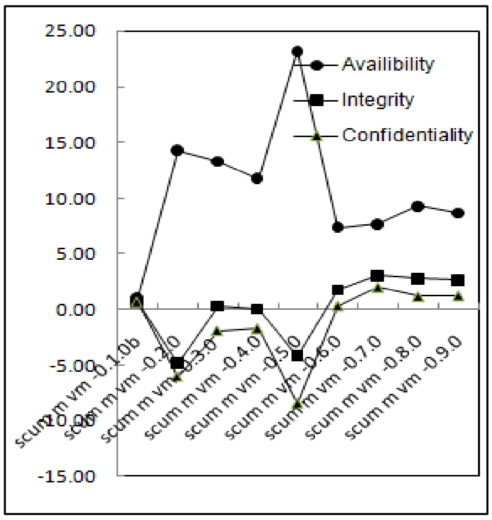

Fig. 3: Plot Of Computed Security Attributes Values For Scum MVM Versions.

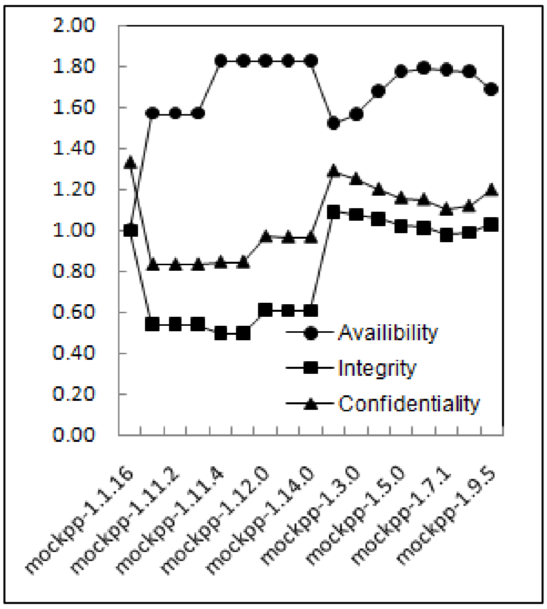

Fig. 4: Plot of Computed Security Attributes Values For Mockpp Versions.

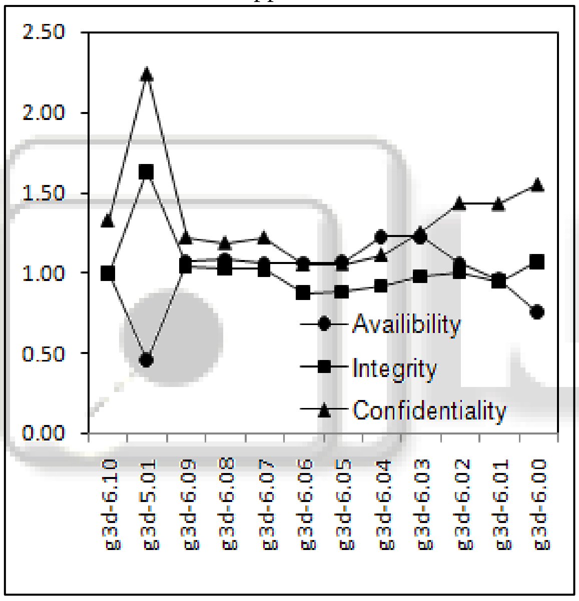

Fig. 5: Plot of Computed Security Attributes Values For G3d Versions.

The same trend is seen with increases in confidentiality, integrity and availability to its predecessors. We have twelve versions of g3d software, after five versions software is showing mature result which is maintain in next seven versions with increases in confidentiality and integrity but slight decrease in availability. Last few versions CIA (confidentiality, integrity and availability) have been increase as compare to with first few versions. For scumm software we have nine versions, after first six versions software is showing mature result which is maintain in next three versions with stable in confidentiality, integrity and increase in availability. Last few stable versions CIA have been increase as compare to with first few versions.

VI. Conclusion

Security is one of the attribute of quality so the same guidelines that are being used to develop the quality metric or model can be used to develop the security metric or model. This research work started with this concept and has successfully developed a security model SMOOD which assess the security of a design. SMOOD model are tested on five software with their all releases we can conclude that results showing by SMOOD model are matching the expected trend and our hypothesis that is in first few releases software is going under lots of changes, even software architecture have been change and after a mature version of software is comes in market which keep the same trend for its next releases. Security of these mature releases is high than its predecessors. As the hypothesized relationships are empirically validated, so the proposed security model MOOD) can be used during the early stages of software development to improve the ultimate security of software products. Thus proposed modeling method could capture the behavior of software systems by use of SMOOD model.

References

[1] Jagdish Bansia, “Hierarchical Model for Object-Oriented Design Quality Assessment”, 0098-5589/02/$17.00©, IEEE Transaction on Software Engineering Vol.28 No. 1 pp. 4-17, 2002.
[2] N.Fenton and B.Kitchenham, “Validating software measures, Journal of Software Testing, Verification and Reliability, Vol. 1, No. 2, pp. 27-42, 1990.
[3] N.Fenton, “Software metrics: Thgeory tools and validation”, Software Engineering Journal, pp. 65-78, 1990.
[4] Agrawal and R.A. Khan, “An Algorithm to Measure Attribute Vulnerability Ratio of an Object Oriented Design”, International Journal of Recent Trends in Engineering, Vol. 2, No. 3, Nov. 2009.
[5] Bandar Alshammari , Colin Fidge and Diane Corney, “Security Metrics for Object-Oriented Class Designs”, QSIC 2009 Proceedings of : Ninth International Conference on Quality Software , August 24-25, 2009, Jeju, Korea.
[6] Chi-Ming Chung, Chun-Chia Wang, Ming-Chi Lee, “Class Hierarchy Based Metric for Object-Oriented Design”, IEEE Xplore pp. 986-991 1994.
[7] Yonghee Shin, Laurie Williams, “Is Complexity Really the Enemy of Software Security?” 978-1-60558-321-1/08/10. 2008 ACM pp. 47-50.
[8] Istehad Chowdhury, Mohammad Zulkernine “Can Complexity, Coupling, and Cohesion Metrics be used as Early Indicators of Vulnerabilities? “, 978-1-60558-638-0/10/03 ACM 2010.
[9] S.P. Kadam, S. D. Joshi, A. V. Dhaigude, “New approach for predicting vulnerability at design stage for object oriented design”, International Journal of Computer Science And Technology, Vol. 2, Issue 3, 2011.
[10] G. McGraw, “Software Security”. IEEE Security &Privacy, IEEE, vol.2, pp.80-83, 2004.
[11] J.Viega and G. McGraw, “Building Secure Software”, Addition Wesley, 2005.
[12] Devpriya Soni, “A Framework for Validation of Object-Oriented Design Metrics”, ISSN 1947-5500 (IJCSIS) International Journal of Computer Science and Information Security, Vol. 6, No.3, pp. 46-55, 2009