Object Oriented Software Security in Design Phase

Mohammad Ubaidullah Bokhari, Shams Tabrez Siddiqui and Hatem Hammatta

September 13, 2021

Object Oriented Software Security in Design Phase

[1]Dr. Mohammad Ubaidullah Bokhari, [2]Shams Tabrez Siddiqui, [3]Hatem S. A. Hamatta
[1,2]Dept. of Computer Science, Aligarh Muslim University, Aligarh, India
[3]Dept. of Applied Sciences, Al-Balqa Applied University, Jordan

Abstract
Object oriented design and development has become popular in today's software development environment. As development of object oriented software is raising security problem is also increasing. The software security focuses on the effort and cost spent in lateral phases is much greater than the initial phase of software development process. It is understandable that most of the software is not well designed with respect to security concern. Therefore, to secure the software after deployment is not only costlier but difficult too. The aim of the paper is to study better way of design consideration to secure the object oriented software and discuss the tools which are required for the secure software development. This paper will provide an opportunity to understand the requirement for developing a technique at design phase.

Keywords
Object Oriented, Software Security, Design Phase, Deployment, And Secure Software Development

I. Introduction

With increasing globalization, use of information system, extended enterprises, and the ubiquitous technologies that make it all possible, the need for information security is taking on greater urgency than ever before. Software security vulnerabilities arise from a number of poor development practices, new mode of attacks, mis-configuration, unsecured links between systems, and poor design. Software vulnerability is a fault in the specification, implementation, or configuration of a software system whose execution can violate explicit or implicit security policy [1] .The software vulnerabilities can be attributed to several causes. Some bugs, like buffer overrun in C and C++, poor computer language, which can be avoided by switching to a safer language. However, safer languages alone can not be preventing security bugs [2].

Security is not a widely accepted feature, it may be considered differently by different people. Key things about security are that we must understand the phrase: Secure against what and from whom? Security can be better understood by identifying the goals. What is it we trying to protect? How can we protect what we want? From whom we are protecting it? [9]. Software Security is a measure of the system's ability to resist unauthorized usage while still providing its services to legitimate users. An attempt to breach security is called an attack and can take a number of forms [3]. Software security is the term largely associated with the process of producing reliable, stable, bug and vulnerable free software. Security is fast becoming one of the most important quality attribute in Software Engineering. Critical information and transactions need to be safeguarded against malicious users and attackers. Copyright protected Commercial Off-the Shelf Components (COTS) may be used in software development [4]. The security landscape has dramatically changed in the last few months. Earlier security was considered as an afterward thought, which can lead to design level security flaws because the primary goal of the development team to deliver a functional system rather than a secure one. Which causes more affords and cost to maintain the software throughout it life period.

"Finding and fixing a software problem after delivery is often 100 times more expensive than finding and fixing it during the requirements and design phase"[5]. In order to maintain the expense software security needs to be considered from very beginning of the software development lifecycle [6,25,8]. It is always better to design a secure system from scratch than try to add security to an existing one [9]. It is widely accepted that the quality of software product is depend upon the design time. Software security defects are mostly born during the implementation time, and its remedy are costlier if its security aspects are not considered in the design phase.

A systematic approach of using absolute attention to security during the design phase prevents expensive redesign and yields substantial benefits during all later phases of the SDLC. Design considerations include both architectural issues at the system level and at the individual component level. At the system level, we are interested in techniques that help us reduce our software's attack surface, and better understand how potential threats might impact our design choices. At the component level, we are interested in deciding how best to implement each module [10]. Requirement is the considered as foundation stone on which the entire software can be built. The failure and success of any software can be depending upon the quality of requirement. In recent survey it is observed that about 71% of the software is not completed due to poor requirement [7, 10].

Software development team use design time metrics to make risk management decision when designing, implementing, and building software and related security mechanism [5].

Object-Oriented technologies greatly influence software development and maintenance through faster development, cost saving and quality improvement and thus have become a major trend for methods of modern software development and system modeling [11]. Class, object, method, message, instance variable, and inheritance are the basic concept of the object-oriented technology [12]. Object oriented metrics are mainly measures of how these constructs are used in designed process. Classes and methods are the basic constructs for object-oriented technology. The amount of function provided by object-oriented software can be estimated based on the number of identified classes and metrics or its variables.

Object-Oriented design measurement is one of the most important parts in secure software development process. Metrics is a set of standards against which one can measure the effectiveness of object-oriented analysis techniques in the design phase. There is number of metrics for object-oriented design (MOOD) methods available which refers to basic structural mechanism of the object-oriented paradigm as Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF) for Encapsulation, Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF) for Inheritance, Polymorphism Factor (PF) for Polymorphism and Coupling Factor (CF) for message passing [13].

II. Design Consideration: A Better Way to Secure Object Oriented Software

Designing secure software is straightforward depending upon the quality of requirement documents. However, even with the best requirements, software design is still a challenging activity and it

should be performed with utmost care and clear goals. The overall goals for designing a secure software process are:

A. Identify the Assets (Components) Need for Security

In the design phase securing the overall components of the system is the prime concern. Identify each asset in the system where it is either in the form of information that the organization possesses, location that the organization controls, or computerized activities that the organization performs needs security carefully at design level [14,15]. Examples of information assets that need security are:

1. Data/Information collections such as databases, data files, procedures, information archives, digital records.
2. Software assets such as application software and system software and custom software.
3. Physical assets, including computers, communication equipment, storage media and even some facility equipment.

Some components needs protection other components will provide protection in the form of validation, authentication, authorization, or cryptographic routines. The security engineer should focus some extra attention on later components. For every components that implements security function must be carefully consider how these function will be implemented, reviewed and tested. Testing of components for its security aspects is a critical task that needs a special attention as early as possible in the development life cycle.

B. Information Asset Classification

To determine the measures required to adequately secure an information asset, the asset must be classified. Classification based on at least one of the three primary information security criteria, confidentiality, integrity, and availability [24]. Criteria for classifying information include:

1. Confidentiality

Confidentiality refers to the sensitivity and the access controls required for protecting the information. Confidentiality is defined in terms of:

- Confidential: Access is restricted to a specific list of people.
- Sensitive: Access and use of the information must be protected from routine disclosure and is restricted to specific uses only.
- Public: Where the resources are publicly accessible.

Access control is a primary component of confidentiality. Who can access to the asset? Who can modify/manipulate the information? How should the information be stored? This is controlled by assigning access rights for individuals, groups, and the public.

2. Availability

This is a measure of criticality. How important is it that the information asset is accessible to the authorized constituent? Is it a single instance or is a backup available? Availability is measured based on reliability and timely access to the asset.

3. Integrity

Integrity refers to maintaining the correctness and consistency of the data. Some integrity checking is made possible by specifying the data type of an item. Integrity is defined in terms of value: high, medium or low. As this is often a subjective valuation, justification may be required for assigning a value classification if the rationale is not obvious or is questionable.

C. Create an Architecture Overview

In this aspect we have to model the system either by DFD or by UML diagram. From these diagrams we can identify the entry points of the system as data source, application programming interfaces and the user interface itself. A broad architecture overview specifies design choice in product creation. An architecture overview provides [16]

- Focus on customer and business
- Direction and guidance to the project team
- Insight in important choices and risks
- Attention for issues that go beyond organizational entities

D. Perform Threat Analysis

Threat modeling is a technique of assessing and documenting a system's security risk. Threat models are methodically developed in order to understand the techniques such as entry point identification, privilege boundaries and threat trees to a system from malicious users [17]. Threat modeling allows development teams to anticipate attacks by understanding how an adversary chooses targets (assets) locate entry points and conduct an attack. To understand the potential threats at any entry points we must have to list all the threats. In these aspects we have to arrange a brainstorming meeting and discuss about the proposed threats such as confidentiality, Access of users privilege, tampering etc. threat analysis can be performed as under:

- Identify the threats: To understand the potential threats at any entry points we must have to list all the threats. In these aspects we have to arrange an brainstorming meeting and discuss about the proposed threats such as confidentiality, Access of users privilege, tampering etc. and those which have higher priority must be attended at most priority
- Categorize the threats using the STRIDE model (Spoofing, Tampering, Repudiation, Information disclosure, Denial of service, and Elevation of privilege)
- Rank the threats using the DREAD categories (Damage potential, Reproducibility, Exploitability, Affected users, and Discoverability).
- Develop threat mitigation strategies for the highest ranking threats

E. Design Techniques that Mitigate Risks

The security risks of any kind can be mitigated either in design phase or in other phases. The earlier mitigation of risks save affords and cost as compared to the later phases. Design is the better place to initiate security thinking and reduce the expenses which would be compounded at later stages. There are many design principles, such are:

1. Principle of Least Privileges

All code should run with the lowest permission levels it needs to perform its assigned task and defense in depth. It is a good practice to use design techniques recommended by the development platform

2. Role of the Programming Language

Languages that target a virtual machine such as Java and C# reduce the risk of buffer overflow vulnerabilities [18]. Strongly typed languages reduce the risk of integer overflow and much data-oriented vulnerability without requiring the implementation of additional protective measures.

F. Security Test Plan

Since software security is a non functional requirement, therefore, its test is separate from the functional, performance, and other testing concerns. Software security testing needs some special tools and persons that are beyond the normal function of a quality assurance. Security testing needs to be highly visible and verifiable by external audits which specify what should not occur when a user applies a series of inputs. Security test plan can be directly derived from the result of threat analysis. A security test plan must first be concerned with testing the threats and attacks. As soon as testing starts, the test plan expanded to cover additional attacks and conditions that were not discovered during the threat modeling process. The process of security testing always uncovers some information about the application that was not discovered in the threat model. The main objective for security test plan of an operational system is to identify potential vulnerabilities and subsequently repairing [19].

G. Class Design Consideration

The attacking surface of a system can be reducing by designing a class using object oriented design principles, which includes:

1. Prevent inheritance where it is not required and limit who can call them.
2. Restrict class and member access modifier by allowing fewer public interfaces in code, smaller the attack surface, the easier to protect. Use the private access modifier wherever possible, protected access modifier if the member should be accessible to derived class, and Internal access modifier only if the member should be accessible to other classes in the same programs.

H. Plans for Incident Response

Delivering a 100% secured software is unrealistic. An attacker targeted the software for exploitation and all software, given enough effort, can eventually be exploited. Ensure you have a plan for dealing with security issues when they occur. Incident response procedures need to be documented and understood. Future plan for security incidents is generally a common mistake committed by the organization which causes unexpected delays to current development tasks in order to respond to a critical customer security issue.

I. Not to Trust on Input

All the input to the application should be validated and checked by its type, length, format, and range. Any input should be trusted. An attacker can attempt by using SQL injection, Cross-site scripting, and other injection attack and exploit the application's vulnerabilities.

J. Use Private Default Constructors to Prevent Unwanted Object Instantiation

A class with a public constructor can be instantiated. The class's constructor should be private if it is not designed for instantiation is a one that contains only static methods and properties. The private constructors are enforced at compile time only.

K. Use of Cryptography for Security Policy Evaluation

Strong names signatures provide authentication for code and cannot be spoofed which provide cryptographically strong evidence for code access security policy evaluation. This allows administrators to grant permissions to specific assemblies. For example, the public key component of a strong name is often used to represent a particular organization.

L. Use of Biometric identity

For strong identity of authenticated user a system must have biometric features like retina recognition, fingerprint identification, voice recognition system etc. a large number of face recognition techniques use face representation found by unsupervised statistical method [20]. These techniques have been applied to 3D object recognition [21], sign recognition [22], and autonomous navigation [23] among many other image analysis problems. It prevents the malicious user to attack the vulnerabilities of an application.

III. Required Tools

Tools required for developing a technique at design phase of secure software life cycle are:

A. GUI Design Studio

GUI Design Studio is a graphical user interface design tool developed by Microsoft that can use be to rapidly create demonstration prototypes without any coding or scripting. Process Modeling and Management Tools: Provide links to other tools that provide support to defined process activities.

B. Microsoft Baseline Security Analyzer (MBSA 2.0)

Microsoft Baseline Security Analyzer (MBSA) is an easy-to-use tool designed for the IT professional that helps small- and medium-sized businesses determine their security state.. Improve security management process by using MBSA to detect common security mis-configurations and missing security updates on your computer systems.

C. Enterprise Architect

Enterprise Architect's UML® tools are flexible and powerful UML modeling tools for the Windows platform. An object oriented UML and business analysis tool for the full development life-cycle, Enterprise Architect provides the competitive edge for software development, project management, requirements management and business analysis.

D. QAInspect

QAInspect allows Quality Assurance professionals to execute comprehensive security tests without security expertise and within the IBM test environment. QAInspect is designed to minimize the time required to execute security tests, enabling QA professionals to maintain focus on functional and performance testing.

E. Practical Threat Analysis Tool

PTA (Practical Threat Analysis) is a software technology and a suite of tools that enable users to find the most beneficial and cost-effective way to secure computerized systems and applications according to their specific functionality and environment.

F. Compuware QA Director

A powerful new capability with a unique visual interface that allows the QA team to adjust testing plans dynamically and immediately understand the impact on test coverage.

IV. Conclusion

The software security focuses on the effort and cost spent in lateral phases is much greater than the initial phase of software development process. It is clear that most of the software is not

well designed with respect to security concern. Therefore, to secure the software after deployment is not only costlier but difficult too. In this paper we study different ways of considering design for secure software development and tools required for secure software development. This paper will provide an opportunity to understand the requirement for developing a technique at design phase.

V. Future work

To develop a technique at design phase of secure software life cycle to implement the security aspect which will focus on issues related to feasibility, efficiency, scalability of software security techniques. To examine the implication and possibilities for software metrics in design phase. The system will consists of following components:

- A complete mitigation technique of all the software security holes found in design phase of SDLC.
- Development of an object-oriented software security metrics in design phase.
- A sequence of design methods, which include the direction of different types of programming design.
- An application to allow customers to run and mark the problem created during its application.
- A prototyping that can be used for problem analysis with respect to security concern.
- Development of a software security design tool.

References

[1] Krsul.I.V,"Software Vulnerability Analysis", Ph.D thesis, Purdue University, pp. 1-188, 1998

[2] Hao Chen, David Wagner,"MOPS: an Infrastructure for Examining Security Properties of Software", CCS'02, ACM Press, 2002.

[3] L Bass, P Clements, R Kazman,"Software Architecture in Practice", Second Edition, SEI Series in Software Engineering, 2003.

[4] Pankaj Goyal,"CS497- Security Engineering and Software Engineering", Department of Computer Science & Engineering, IIT, Kanpur, India, 2005.

[5] Bokhari, M. U., Siddiqui, Shams T.,"A Comparative study of software requirement tools for secure software development", in BVICAM'S International journal of IT(BIJIT), Vol. 2, No. 2, pp. 1-8, July-December, 2010.

[6] Wnag, H, .C. Wang,"Taxonomy of Security Consideration and Software Quality", Communication of the ACM, Vol. 46, Issue 6, pp. 75-78, ACM Press 2003.

[7] Bokhari, M. U, Siddiqui, Shams T.,"Metrics for Requirement Engineering and Automated Requirement Tools", Proceedings of the 5th National conference; INDIACom-2011, New Delhi.

[8] Jurjens, J,"Using UMLSec, and goal trees for Secure Systems Development", Proceedings of the 2002 ACM Symposium on Applied Computing, pp. 1024-1030, ACM Press, 2002.

[9] John Viega, Gary McGraw,"Building Secure Software", Addison Wesley, 2005

[10] John Pescatore,"FirstTake FT-23-5794", Gartner Research, July 2004.

[11] Chu, C. Williams., Lu, Chih-Wei., Chang, Chih-Hung., Chung, Yeh-Ching., Huang, Yueh Min., Xu, Baowen.,"Software Maintainability Improvement : Integrating Standards and Models", IEEE Proceedings of the 26th Annual International Computer Software and Application Conference, 2002.

[12] Stephen K Kan,"Metrics and Models in Software Quality Engineering", 2nd edition, Pearson Education, 2003.

[13] Umit Karakab, Sencer Sultanoolu,"Object Oriented Metrics", 1998.

[14] Sindre G, Firesmith D, Opdahl AL,"A reuse-based approach to determining security requirements", In: Proceedings of the 9th Intl. workshop on requirements engineering: foundation for software quality (REFSQ'03), Klagenfurt, Austria, 2003.

[15] Guttorm Sindre, Andreas L. Opdahl,"Eliciting security requirements with misuse cases", Journal of Requirement Engineering, Springer Verlog, Vol. 10, pp. 34-44, January 2005.

[16] Gerrit Muller,"How to Create an Architecture Overview", Embedded System Institute, 9th August 2006.

[17] Scott W. Ambler,"Introduction to Security Threat Modeling", Agile Modeling, April 2006.

[18] Jason Taylor,"Web services Risk Assessment and Recommendation", Security Innovation Commercial Tools Division 1318 S, Babcock Street, Melbourne, 2003/2004.

[19] John Wack, Miles Tracy, Murugiah Souppaya,"Guideline on Network Security Testing", NIST Special Publication 800-42, US Dept. of Commerce, October 2003.

[20] Issam Daghar, Rabih Nachar,"Face Recognition Using IPCA-ICA Algorithm", IEEE Transaction on Pattern Analysis and Machine Intelligence, Vol. 28, No. 6, pp. 996-1000, June 2006.

[21] H. Murase, S.K. Nayar,"Visual Learning and Recognition of 3D Objects from Appearance", Intl. Computer Vision, Vol. 14 No. 1, pp-5-24, Jan 1995.

[22] Y. Cui, J. Weng,"Appearnce-Base Hand Sign Recognition from Intensity Image Sequences", Computer Vision and Image Understanding, Vol. 78, pp. 157-176, 2000.

[23] S. Chen, J. Weng,"State-Based SCHOSLIF for Indoor Visual Navigation", IEEE Transaction, Neural Networks, Vol. 11, No. 6, pp. 1300-1314, 2000.

[24] Barry Boehm, Victor R. Basili,"Software Defect Reduction Top 10 List", Software Management, pp. 135-137, January 2001.

[25] Devanbu, P.T., S. Stubblebine,"Software Engineering for Security a Roadmap", Proceedings of the Conference on the future of Software Engineering, pp. 227-239, ACM Press 2000.

Dr. Mohammad Ubaidullah Bokhari is currently working as Chairman and Associate Professor, Department of Computer Science, AMU, Aligarh. He has published more than 72 research papers in different reputed journals and conference proceedings. He has also authored 5 books on different fields of Computer Science. His current research interests are Requirement Engineering, cryptography, Software Reliability, Wireless Network Security and Database.

Shams Tabrez Siddiqui received his B.Sc and MCA degree from Aligarh Muslim University, Aligarh, India in 2003 and 2007. He is pursuing Ph.D in Software Requirement Engineering from AMU, Aligarh. He is also working as a counselor for IGNOU. His research interest includes Software Engineering, Requirement Engineering and Software Security. He is a member of the IEEE Computer Society, IAENG and IACSIT.

Hatem S. A. Hamatta received his B.Sc. degree in Computer Science from Yarmouk University, Irbid, Jordan, in 2003, the M.Sc. degree in Computer Science from University of Jordan, Amman, Jordan, in 2006. He is currently pursuing his research work of PhD Degree in Wireless Network Security in Aligarh Muslim University, Aligarh, India. Since 2006, he has been with the faculty of the Department of Applied Sciences at Al-Balqa` Applied University/Aqaba University College, Aqaba, Jordan; where he is currently a Senior Lecturer. His research interests include Wireless Network Security, Mobile Ad Hoc Networks, Intrusion Detection Systems and Security Design Issues.